Regression Analysis on Spotify Song Attributes - MATH564

Lakshmi Sindhu Pulugundla, Lasya Priya Thota, Kaustubh Dangche

Introduction

Problem Statement: This analysis aims to understand the relationship between various song attributes and their total playback time (`msPlayed`) on Spotify. The objective is to identify which factors significantly influence playback duration and uncover patterns or trends in user engagement.

Project Goal: To develop a regression model that predicts song playback time (`msPlayed`) based on:

- **Continuous attributes** such as `danceability`, `energy`, and `tempo`.
- **Categorical attributes** such as `genre` and `artistName`.

Expected Results/Outcomes:

- A regression model explaining significant variations in playback time.
- Insights into how song attributes influence user engagement.
- Identification and remediation of model issues to improve accuracy and reliability.

Dataset Overview: The Spotify Song Attributes dataset includes 10,080 records with 22 variables describing various song features. Key attributes include:

- **Response Variable:** `msPlayed` (total playback duration in milliseconds).
- **Continuous Variables:** `danceability`, `energy`, `tempo`, `loudness`.
- **Categorical Variables:** `genre`, `artistName`.

Preprocessing and cleaning were required to handle missing values, inconsistencies, and outliers to ensure the quality of the insights and predictions.

Project Workflow

Part 1: Initial Analysis

This phase established the groundwork for model development, including data preparation, feature selection, model specification, and statistical analysis.

Dataset Selection and Exploration

1. The Spotify dataset was chosen for its real-world relevance and inclusion of both continuous and categorical variables.
2. **Response Variable:** `msPlayed`.
3. **Predictors:**
   - Continuous: `danceability`, `energy`, `tempo`.
   - Categorical: `genre`, `artistName`.

Data Cleaning

1. **Handling missing values:**
   - Missing values in `danceability`, `energy`, and `tempo` were imputed using KNN imputation to maintain data integrity.
2. **Cleaning categorical variables:**
   - `genre` and `artistName` were standardized (e.g., lowercase conversion, grouping rare categories as "Other").
3. **Outlier handling:**
   - Extreme values in continuous variables (`danceability`, `energy`, and `tempo`) were capped using the IQR method.
4. **Normalization:**
   - Continuous variables were scaled to a 0-1 range for comparability.

Model Specification

1. A multiple linear regression model was defined:
   - **Response Variable:** `msPlayed`.
   - **Predictors:** `danceability`, `energy`, `tempo`, `genre`, and `artistName`.
2. The model was fitted, and overall significance was evaluated using F-statistics.

Statistical Significance and Interpretation

1. **Regression Coefficients:** Interpretation of predictors (significant and non-significant) was completed.
2. **Goodness-of-Fit Metrics:** Model $R^2 = 0.0841$ and Adjusted $R^2 = 0.04247$ indicated room for improvement, highlighting the need for diagnostics and refinement.

Part 2: Regression Diagnostics

This phase investigates model assumptions and identifies potential issues to address for improved reliability.

Assumptions and Potential Issues

1. **Heteroscedasticity:** Diagnostic plots are being generated and analyzed.
2. **Multicollinearity:** Variance Inflation Factor (VIF) is being assessed.
3. **Influential Points:** Cook's Distance and leverage values are being calculated.

Status: Diagnostics are partially completed, with findings documented for heteroscedasticity and multicollinearity. Influential point analysis is in progress.

Part 3: Remediation and Refinement

This phase addresses issues identified in the diagnostic phase.

Remediation

1. **Proposed Remediation Techniques:**
   - Logarithmic transformations for heteroscedasticity.
   - Weighted Least Squares to address unequal variance.
   - Polynomial or interaction terms for non-linear relationships.
   - Adjusting for influential points based on diagnostics.
2. **Re-fitting and Comparison:**
   - The model will be refitted after applying remediation techniques, with improvements and limitations documented.

Summary and Findings

This final phase summarizes the analysis, discussing key findings, diagnostic issues, and the effectiveness of remediation techniques.

Let's Dive into the Implementation part now - # **Part 1: Initial Analysis** ## 1. Data Selection and Exploration - We chose a realworld dataset containing both continuous and Categorical Variables. Now let's focus on Exploration for which we will first have to-

1.1 Load the necessary libraries and dataset:

```
# Load necessary libraries
library(dplyr)  # For data manipulation
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)  # Optional, for visualization if needed
#install.packages("VIM") # KNN imputation
library(VIM)
```

```
## Warning: package 'VIM' was built under R version 4.4.2

## Loading required package: colorspace

## Loading required package: grid

## VIM is ready to use.

## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues

##
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':
##
##     sleep
```

```r
library(corrplot)
```

```
## corrplot 0.94 loaded
```

```r
# Load the Spotify attribute dataset
data <- read.csv("./data/Spotify_Song_Attributes.csv")
# Create a copy of the spotify attribute dataset
spotify_data <- data
# View the first few rows to understand the structure
head(spotify_data)
```

```
##                                                                   trackName
## 1                                                                  "Honest"
## 2 "In The Hall Of The Mountain King" from Peer Gynt Suite N°1, Op. 46
## 3                                                        #BrooklynBloodPop!
## 4                                                                       $10
## 5                                                  (I Just) Died In Your Arms
## 6                                                              (L)only Child
```

```
##                  artistName msPlayed             genre danceability energy key
## 1              Nico Collins   191772                          0.476  0.799   4
## 2 London Symphony Orchestra  1806234 british orchestra        0.475  0.130   7
## 3                      SyKo   145610        glitchcore        0.691  0.814   1
## 4              Good Morning    25058  experimental pop        0.624  0.596   4
## 5              Cutting Crew  5504949        album rock        0.625  0.726  11
## 6               salem ilese  2237969             alt z        0.645  0.611   8
##   loudness mode speechiness acousticness instrumentalness liveness valence
## 1   -4.939    0      0.2120       0.0162         0.00e+00   0.2570   0.577
## 2  -17.719    1      0.0510       0.9160         9.56e-01   0.1010   0.122
## 3   -3.788    0      0.1170       0.0164         0.00e+00   0.3660   0.509
## 4   -9.804    1      0.0314       0.4750         2.03e-01   0.1190   0.896
## 5  -11.402    0      0.0444       0.0158         1.69e-04   0.0625   0.507
## 6   -5.925    0      0.1370       0.2900         2.05e-05   0.2370   0.645
##     tempo           type                     id
## 1 162.139 audio_features 7dTxqsaFGHOXwtzHINjfHv
## 2 112.241 audio_features 14Qcrx6DfjvcjOH8oV8oUW
## 3 132.012 audio_features 7K9Z3yFNNLv5kwTjQYGjnu
## 4 120.969 audio_features 3koAwrM1ROOTGMeQJ3qt9J
## 5 124.945 audio_features 4ByEFOBuLXpCqvO1kw8Wdm
## 6 157.475 audio_features 22lJaG2yxlSjIwdUIddcFk
##                                    uri
## 1 spotify:track:7dTxqsaFGHOXwtzHINjfHv
## 2 spotify:track:14Qcrx6DfjvcjOH8oV8oUW
## 3 spotify:track:7K9Z3yFNNLv5kwTjQYGjnu
## 4 spotify:track:3koAwrM1ROOTGMeQJ3qt9J
## 5 spotify:track:4ByEFOBuLXpCqvO1kw8Wdm
## 6 spotify:track:22lJaG2yxlSjIwdUIddcFk
##                                                 track_href
## 1 https://api.spotify.com/v1/tracks/7dTxqsaFGHOXwtzHINjfHv
## 2 https://api.spotify.com/v1/tracks/14Qcrx6DfjvcjOH8oV8oUW
## 3 https://api.spotify.com/v1/tracks/7K9Z3yFNNLv5kwTjQYGjnu
## 4 https://api.spotify.com/v1/tracks/3koAwrM1ROOTGMeQJ3qt9J
## 5 https://api.spotify.com/v1/tracks/4ByEFOBuLXpCqvO1kw8Wdm
## 6 https://api.spotify.com/v1/tracks/22lJaG2yxlSjIwdUIddcFk
##                                                         analysis_url duration_ms
## 1 https://api.spotify.com/v1/audio-analysis/7dTxqsaFGHOXwtzHINjfHv      191948
## 2 https://api.spotify.com/v1/audio-analysis/14Qcrx6DfjvcjOH8oV8oUW      150827
## 3 https://api.spotify.com/v1/audio-analysis/7K9Z3yFNNLv5kwTjQYGjnu      145611
## 4 https://api.spotify.com/v1/audio-analysis/3koAwrM1ROOTGMeQJ3qt9J       89509
## 5 https://api.spotify.com/v1/audio-analysis/4ByEFOBuLXpCqvO1kw8Wdm      280400
## 6 https://api.spotify.com/v1/audio-analysis/22lJaG2yxlSjIwdUIddcFk      144468
##   time_signature
## 1              4
## 2              4
## 3              4
## 4              4
## 5              4
## 6              3
```

Inference: The dataset is successfully loaded into R, and a copy (`spotify_data`) is created to preserve the original data. We can now inspect the first few rows to understand its structure.

1.2 View Dataset Structure

```r
# View the structure of the data
str(spotify_data)
```

```
## 'data.frame':    10080 obs. of  22 variables:
##  $ trackName       : chr  "\"Honest\"" "\"In The Hall Of The Mountain King\" from Peer Gynt Suite N°
##  $ artistName      : chr  "Nico Collins" "London Symphony Orchestra" "SyKo" "Good Morning" ...
##  $ msPlayed        : int  191772 1806234 145610 25058 5504949 2237969 441335 70589 120005 107407 ...
##  $ genre           : chr  "" "british orchestra" "glitchcore" "experimental pop" ...
##  $ danceability    : num  0.476 0.475 0.691 0.624 0.625 0.645 0.663 NA 0.792 0.759 ...
##  $ energy          : num  0.799 0.13 0.814 0.596 0.726 0.611 0.904 NA 0.511 0.699 ...
##  $ key             : num  4 7 1 4 11 8 7 NA 2 0 ...
##  $ loudness        : num  -4.94 -17.72 -3.79 -9.8 -11.4 ...
##  $ mode            : num  0 1 0 1 0 0 1 NA 1 0 ...
##  $ speechiness     : num  0.212 0.051 0.117 0.0314 0.0444 0.137 0.0857 NA 0.0409 0.0307 ...
##  $ acousticness    : num  0.0162 0.916 0.0164 0.475 0.0158 0.29 0.000708 NA 0.124 0.202 ...
##  $ instrumentalness: num  0.00 9.56e-01 0.00 2.03e-01 1.69e-04 2.05e-05 2.89e-01 NA 9.04e-05 1.31e-0
##  $ liveness        : num  0.257 0.101 0.366 0.119 0.0625 0.237 0.341 NA 0.14 0.443 ...
##  $ valence         : num  0.577 0.122 0.509 0.896 0.507 0.645 0.675 NA 0.111 0.907 ...
##  $ tempo           : num  162 112 132 121 125 ...
##  $ type            : chr  "audio_features" "audio_features" "audio_features" "audio_features" ...
##  $ id              : chr  "7dTxqsaFGHOXwtzHINjfHv" "14Qcrx6DfjvcjOH8oV8oUW" "7K9Z3yFNNLv5kwTjQYGjnu"
##  $ uri             : chr  "spotify:track:7dTxqsaFGHOXwtzHINjfHv" "spotify:track:14Qcrx6DfjvcjOH8oV8o
##  $ track_href      : chr  "https://api.spotify.com/v1/tracks/7dTxqsaFGHOXwtzHINjfHv" "https://api.sp
##  $ analysis_url    : chr  "https://api.spotify.com/v1/audio-analysis/7dTxqsaFGHOXwtzHINjfHv" "https:/
##  $ duration_ms     : num  191948 150827 145611 89509 280400 ...
##  $ time_signature  : num  4 4 4 4 4 3 4 NA 4 4 ...
```

Inference:

- The dataset contains **10,080 observations** and **22 variables**, including both continuous and categorical attributes.
- Key continuous variables: `msPlayed`, `danceability`, `energy`, `tempo`, and `loudness`.
- Key categorical variables: `genre` and `artistName`.
- Some columns, such as `track_href`, `uri`, and `id`, are non-informative for analysis and can be removed.
- Missing values are present in critical columns like `danceability`, `energy`, and `tempo` (550 rows each).
- Data types are generally appropriate (e.g., numeric for continuous variables, character for categorical), but some categorical variables require standardization (e.g., `genre` and `artistName`).

1.3 Statistical Summary of Each Column:

```r
# View a summary of each column (e.g., min, max, mean, etc.)
summary(spotify_data)
```

```
##   trackName          artistName           msPlayed             genre
##  Length:10080       Length:10080       Min.   :       0   Length:10080
##  Class :character   Class :character   1st Qu.:  136780   Class :character
##  Mode  :character   Mode  :character   Median :  266288   Mode  :character
```

```
##                                        Mean   :  1519657
##                                        3rd Qu.:  1186307
##                                        Max.   :158367130
##
##   danceability        energy            key            loudness
##  Min.   :0.0000   Min.   :0.0011   Min.   : 0.000   Min.   :-42.044
##  1st Qu.:0.5090   1st Qu.:0.4030   1st Qu.: 2.000   1st Qu.:-10.189
##  Median :0.6230   Median :0.5890   Median : 5.000   Median : -7.218
##  Mean   :0.6025   Mean   :0.5635   Mean   : 5.242   Mean   : -8.685
##  3rd Qu.:0.7140   3rd Qu.:0.7510   3rd Qu.: 8.000   3rd Qu.: -5.336
##  Max.   :0.9760   Max.   :0.9990   Max.   :11.000   Max.   :  3.010
##  NA's   :550      NA's   :550      NA's   :550      NA's   :550
##       mode         speechiness      acousticness    instrumentalness
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0361   1st Qu.:0.0538   1st Qu.:0.0000
##  Median :1.0000   Median :0.0479   Median :0.2450   Median :0.0000
##  Mean   :0.6124   Mean   :0.0785   Mean   :0.3629   Mean   :0.1532
##  3rd Qu.:1.0000   3rd Qu.:0.0819   3rd Qu.:0.6680   3rd Qu.:0.0276
##  Max.   :1.0000   Max.   :0.9660   Max.   :0.9960   Max.   :0.9930
##  NA's   :550      NA's   :550      NA's   :550      NA's   :550
##     liveness         valence           tempo            type
##  Min.   :0.0249   Min.   :0.0000   Min.   :  0.00   Length:10080
##  1st Qu.:0.0962   1st Qu.:0.2370   1st Qu.: 97.57   Class :character
##  Median :0.1190   Median :0.4090   Median :119.82   Mode  :character
##  Mean   :0.1746   Mean   :0.4341   Mean   :119.37
##  3rd Qu.:0.2090   3rd Qu.:0.6140   3rd Qu.:139.78
##  Max.   :0.9640   Max.   :0.9860   Max.   :236.20
##  NA's   :550      NA's   :550      NA's   :550
##       id                uri             track_href        analysis_url
##  Length:10080       Length:10080       Length:10080       Length:10080
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   duration_ms      time_signature
##  Min.   :  10027   Min.   :0.000
##  1st Qu.: 161697   1st Qu.:4.000
##  Median : 194286   Median :4.000
##  Mean   : 202931   Mean   :3.917
##  3rd Qu.: 229526   3rd Qu.:4.000
##  Max.   :4581483   Max.   :5.000
##  NA's   :550       NA's   :550
```

Inference:

- **Response Variable (`msPlayed`)**:
  - Highly variable, with values ranging from **0** to **158,367,130 ms**.
  - Median playback time is **266,288 ms**, with a mean of approximately **1,519,657 ms**, suggesting potential outliers.
- **Continuous Variables**:

  - `danceability`, `energy`, and `tempo`:
    * Distributions appear to range from minimal to maximum values, indicating diversity in song attributes.
    * Missing values (550 rows) may affect modeling and require handling.
  - `loudness`:
    * Negative mean and median values indicate quieter tracks overall, with some louder tracks present.

- **Categorical Variables**:
  - `genre`:
    * A mix of general and specific genres, with missing or empty values recorded.
  - `artistName`:
    * Significant variability, with some names potentially requiring standardization (e.g., handling special characters, whitespace).

- **Other Observations**:
  - Variables such as `id`, `uri`, and `track_href` are identifiers and do not contribute to the analysis.
  - Columns like `type` are constant and can be excluded.

With this understanding of the dataset's structure and attributes, the next step is to identify and select three continuous variables and two categorical variables that are most relevant for modeling. These selected predictors will form the foundation of our regression analysis, guiding feature engineering and model specification.

1.4 Select 3 Continuous and 2 Categorical Variables:

The selection of variables is a critical step in regression analysis, as it determines the predictors that will explain the variability in the response variable `(msPlayed)`.

For continuous variables -

- Attributes with **high variability**.
- Strong **relevance** to playback time.
- Minimal **missing values**.

For categorical variables -

- Predictors with **sufficient representation** in the dataset.
- Potential **influence** on playback time based on domain knowledge.

This approach should ensure that the model captures key **patterns** and **interactions** in the data effectively.

1.4.1 Selection of Variables Let's start with the Continuous Variables:

```r
# Response Variable
response_variable <- "msPlayed"

# Continuous Variables: Selecting based on variability and relevance

# Checking summary statistics of numeric variables to identify candidates
numeric_cols <- sapply(spotify_data, is.numeric)
numeric_summary <- summary(spotify_data[, numeric_cols])

# Displaying summary for analysis
print(numeric_summary)
```

```
##     msPlayed          danceability        energy            key        
##  Min.   :        0   Min.   :0.0000   Min.   :0.0011   Min.   : 0.000  
##  1st Qu.:   136780   1st Qu.:0.5090   1st Qu.:0.4030   1st Qu.: 2.000  
##  Median :   266288   Median :0.6230   Median :0.5890   Median : 5.000  
##  Mean   :  1519657   Mean   :0.6025   Mean   :0.5635   Mean   : 5.242  
##  3rd Qu.:  1186307   3rd Qu.:0.7140   3rd Qu.:0.7510   3rd Qu.: 8.000  
##  Max.   :158367130   Max.   :0.9760   Max.   :0.9990   Max.   :11.000  
##                      NA's   :550      NA's   :550      NA's   :550     
##     loudness            mode         speechiness      acousticness   
##  Min.   :-42.044   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:-10.189   1st Qu.:0.0000   1st Qu.:0.0361   1st Qu.:0.0538  
##  Median : -7.218   Median :1.0000   Median :0.0479   Median :0.2450  
##  Mean   : -8.685   Mean   :0.6124   Mean   :0.0785   Mean   :0.3629  
##  3rd Qu.: -5.336   3rd Qu.:1.0000   3rd Qu.:0.0819   3rd Qu.:0.6680  
##  Max.   :  3.010   Max.   :1.0000   Max.   :0.9660   Max.   :0.9960  
##  NA's   :550       NA's   :550      NA's   :550      NA's   :550     
##  instrumentalness    liveness         valence           tempo       
##  Min.   :0.0000   Min.   :0.0249   Min.   :0.0000   Min.   :  0.00  
##  1st Qu.:0.0000   1st Qu.:0.0962   1st Qu.:0.2370   1st Qu.: 97.57  
##  Median :0.0000   Median :0.1190   Median :0.4090   Median :119.82  
##  Mean   :0.1532   Mean   :0.1746   Mean   :0.4341   Mean   :119.37  
##  3rd Qu.:0.0276   3rd Qu.:0.2090   3rd Qu.:0.6140   3rd Qu.:139.78  
##  Max.   :0.9930   Max.   :0.9640   Max.   :0.9860   Max.   :236.20  
##  NA's   :550      NA's   :550      NA's   :550      NA's   :550     
##   duration_ms      time_signature 
##  Min.   :  10027   Min.   :0.000  
##  1st Qu.: 161697   1st Qu.:4.000  
##  Median : 194286   Median :4.000  
##  Mean   : 202931   Mean   :3.917  
##  3rd Qu.: 229526   3rd Qu.:4.000  
##  Max.   :4581483   Max.   :5.000  
##  NA's   :550       NA's   :550
```

Selected Variables: From the dataset, the following continuous variables stand out for their relevance to playback duration:
1. **Danceability**: How suitable a track is for dancing, with values ranging from 0 to 0.9760 (Mean: 0.6025).
2. **Energy**: A measure of intensity and activity, ranging from 0.0011 to 0.9990 (Mean: 0.5635).
3. **Tempo**: The track's pace in beats per minute, varying between 0 and 236.20 (Mean: 119.37).

Issues Identified:

- **Missing Values**: All three variables have 550 missing values that must be addressed before analysis.
- **Outliers**: Extremely low values in `tempo` (Min: 0) and `energy` (Min: 0.0011) may indicate errors or unusual cases that need review.
- **Scaling**: These variables might require normalization to ensure they contribute equally to the regression model.

Inference: These continuous variables provide meaningful insights into track characteristics and their influence on playback duration. However, careful preprocessing is crucial to handle missing data and outliers effectively, ensuring the reliability of the analysis.

With the continuous variables selected, the next step is to dive into the categorical variables, such as `genre` and `artistName`. These features can reveal trends and patterns that help explain playback duration from a categorical perspective. Let's explore their potential.

But before that, let's load the selected columns into a list

```
continuous_variables <- c("danceability", "energy", "tempo")
```

Now, let's look into the Categorical Variables -

```
# Categorical Variables: Identifying based on frequency distribution and relevance
# Analyzing the unique counts of categorical columns
categorical_cols <- sapply(spotify_data, is.character)
spotify_data %>%
  select(which(categorical_cols)) %>%
  summarise_all(~ length(unique(.)))
```

```
##   trackName artistName genre type   id  uri track_href analysis_url
## 1      4815       2312   524    2 4737 4737       4737         4737
```

```
# Suggested Categorical Variables (based on analysis)
categorical_variables <- c("genre", "artistName")
```

Inference: The code analyzed the unique value counts of all categorical columns in the dataset to identify meaningful predictors. Among the categorical columns: - **`genre`** has 524 unique values, representing various music genres, making it a promising candidate for analysis. - **`artistName`** has 2,312 unique values, capturing a wide range of artists, which may provide insights into playback patterns. - Other columns such as **`type`**, **`id`**, **`uri`**, **`track_href`**, and **`analysis_url`** primarily contain identifiers or URLs with limited analytical relevance. These columns can be excluded from further analysis.

This analysis confirms that **`genre`** and **`artistName`** are the most relevant categorical variables for inclusion in the regression model. Their diversity offers potential for capturing patterns in playback duration, provided these categories are effectively cleaned and standardized.

Having identified **`danceability`**, **`energy`**, and **`tempo`** as the key continuous variables and **`genre`** and **`artistName`** as the critical categorical variables, the next step focuses on data cleaning. This involves handling missing values, addressing potential outliers, and standardizing categorical variables to ensure they are model-ready. Let's proceed with preparing these variables for the regression analysis.

2. Data Cleaning:

2.1 Handling Missing Values

Identifyig Missing Values -

```r
colSums(is.na(spotify_data))
```

```
##        trackName       artistName         msPlayed            genre
##                0                0                0                0
##     danceability           energy              key         loudness
##              550              550              550              550
##             mode      speechiness     acousticness instrumentalness
##              550              550              550              550
##         liveness          valence            tempo             type
##              550              550              550                0
##               id              uri       track_href     analysis_url
##                0                0                0                0
##      duration_ms   time_signature
##              550              550
```

Observation: Columns like `danceability`, `energy`, `tempo`, and others have 550 missing values, while some columns have no missing values.

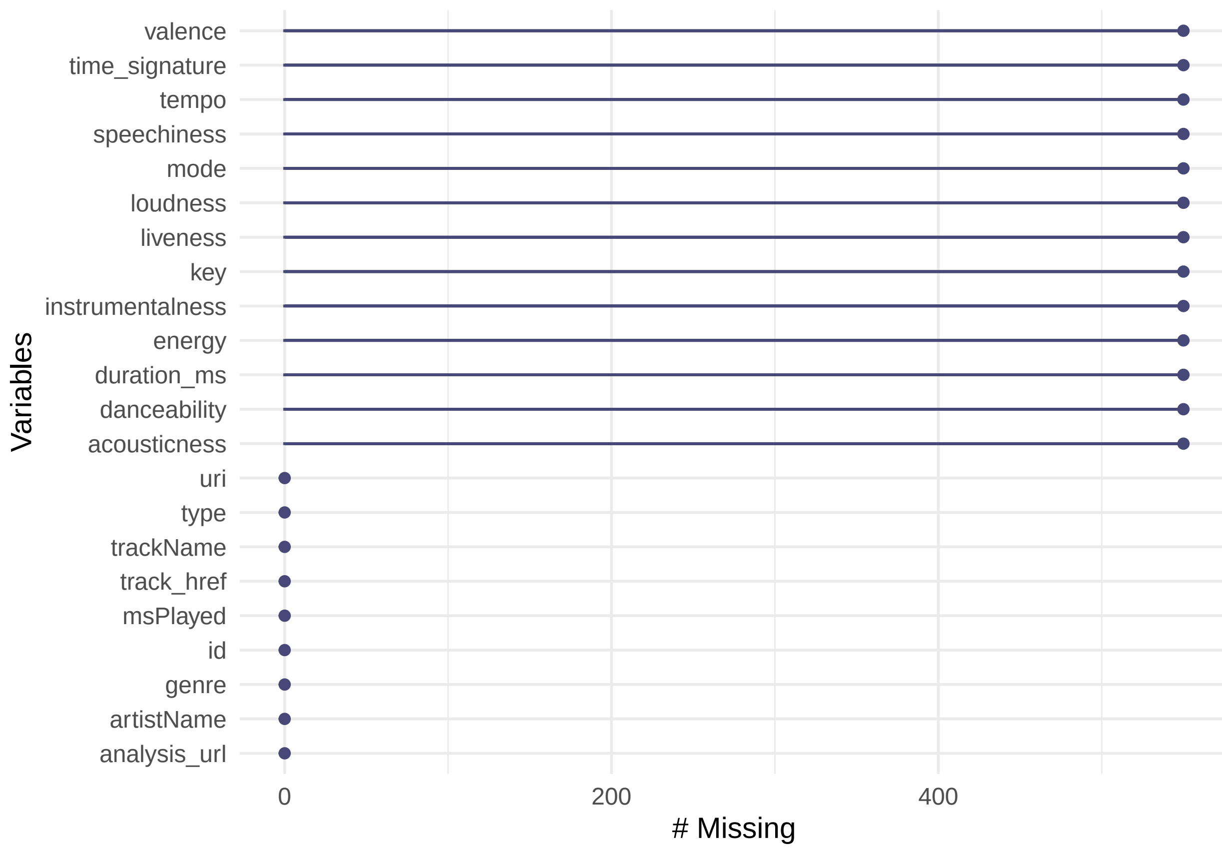
To handle the missing values, let's first try and analyse the missing value summary -

```r
missing_summary <- colSums(is.na(spotify_data)) / nrow(spotify_data) * 100
print(missing_summary)
```

```
##        trackName       artistName         msPlayed            genre
##         0.000000         0.000000         0.000000         0.000000
##     danceability           energy              key         loudness
##         5.456349         5.456349         5.456349         5.456349
##             mode      speechiness     acousticness instrumentalness
##         5.456349         5.456349         5.456349         5.456349
##         liveness          valence            tempo             type
##         5.456349         5.456349         5.456349         0.000000
##               id              uri       track_href     analysis_url
##         0.000000         0.000000         0.000000         0.000000
##      duration_ms   time_signature
##         5.456349         5.456349
```

Let's check for missing data patterns -

```r
library(naniar)
gg_miss_var(spotify_data)
```



Inference: Our selected continuous variables—**danceability**, **energy**, and **tempo**—each have around 550 missing values, accounting for about 5.5% of the dataset. Since these variables are central to our regression analysis, removing rows with missing values would significantly reduce our sample size, potentially affecting the model's reliability.

To address this, we considered different imputation methods: - **Mean or Median Imputation** could fill the gaps but might distort the distribution, especially if the data is skewed. - **K-Nearest Neighbors (KNN) Imputation** leverages similar observations to predict missing values, which helps maintain relationships within the data. - **Regression-Based Imputation** and **Multiple Imputation** are alternatives, but they can add complexity without necessarily improving the outcome for our purposes.

Given the balance between accuracy and practicality, **KNN imputation** is our chosen approach. It allows us to estimate missing values based on similarities in the dataset, preserving both data relationships and sample size for a more robust regression analysis.

Let's proceed with KNN imputation to handle the missing values in our continuous variables.

KNN Imputation for Missing Values:

```r
# Specify the continuous variables with missing values
continuous_vars <- c("danceability", "energy", "tempo")

# Perform KNN imputation on the dataset for the specified columns
spotify_data_imputed <- kNN(spotify_data, variable = continuous_vars, k = 5)

# Check if missing values are handled
colSums(is.na(spotify_data_imputed))
```

```
##        trackName       artistName         msPlayed            genre
##                0                0                0                0
##     danceability           energy              key         loudness
##                0                0              550              550
##             mode      speechiness     acousticness instrumentalness
##              550              550              550              550
##         liveness          valence            tempo             type
##              550              550                0                0
##               id              uri       track_href     analysis_url
##                0                0                0                0
##      duration_ms   time_signature danceability_imp       energy_imp
##              550              550                0                0
##        tempo_imp
##                0
```

```r
# View the structure and summary after imputation
str(spotify_data_imputed)
```

```
## 'data.frame':    10080 obs. of  25 variables:
##  $ trackName       : chr  "\"Honest\"" "\"In The Hall Of The Mountain King\" from Peer Gynt Suite N°
##  $ artistName      : chr  "Nico Collins" "London Symphony Orchestra" "SyKo" "Good Morning" ...
##  $ msPlayed        : int  191772 1806234 145610 25058 5504949 2237969 441335 70589 120005 107407 ...
##  $ genre           : chr  "" "british orchestra" "glitchcore" "experimental pop" ...
##  $ danceability    : num  0.476 0.475 0.691 0.624 0.625 0.645 0.663 0.636 0.792 0.759 ...
##  $ energy          : num  0.799 0.13 0.814 0.596 0.726 0.611 0.904 0.335 0.511 0.699 ...
##  $ key             : num  4 7 1 4 11 8 7 NA 2 0 ...
##  $ loudness        : num  -4.94 -17.72 -3.79 -9.8 -11.4 ...
##  $ mode            : num  0 1 0 1 0 0 1 NA 1 0 ...
##  $ speechiness     : num  0.212 0.051 0.117 0.0314 0.0444 0.137 0.0857 NA 0.0409 0.0307 ...
##  $ acousticness    : num  0.0162 0.916 0.0164 0.475 0.0158 0.29 0.000708 NA 0.124 0.202 ...
##  $ instrumentalness: num  0.00 9.56e-01 0.00 2.03e-01 1.69e-04 2.05e-05 2.89e-01 NA 9.04e-05 1.31e-0
##  $ liveness        : num  0.257 0.101 0.366 0.119 0.0625 0.237 0.341 NA 0.14 0.443 ...
##  $ valence         : num  0.577 0.122 0.509 0.896 0.507 0.645 0.675 NA 0.111 0.907 ...
##  $ tempo           : num  162 112 132 121 125 ...
##  $ type            : chr  "audio_features" "audio_features" "audio_features" "audio_features" ...
##  $ id              : chr  "7dTxqsaFGHOXwtzHINjfHv" "14Qcrx6Dfjvcj0H8oV8oUW" "7K9Z3yFNNLv5kwTjQYGjnu"
##  $ uri             : chr  "spotify:track:7dTxqsaFGHOXwtzHINjfHv" "spotify:track:14Qcrx6Dfjvcj0H8oV8o
##  $ track_href      : chr  "https://api.spotify.com/v1/tracks/7dTxqsaFGHOXwtzHINjfHv" "https://api.sp
##  $ analysis_url    : chr  "https://api.spotify.com/v1/audio-analysis/7dTxqsaFGHOXwtzHINjfHv" "https:
##  $ duration_ms     : num  191948 150827 145611 89509 280400 ...
##  $ time_signature  : num  4 4 4 4 4 3 4 NA 4 4 ...
##  $ danceability_imp: logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ energy_imp      : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ tempo_imp       : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
```

```r
summary(spotify_data_imputed[continuous_vars])
```

```
##   danceability        energy            tempo
##  Min.   :0.0000   Min.   :0.00108   Min.   :  0.00
##  1st Qu.:0.5170   1st Qu.:0.35200   1st Qu.: 99.02
##  Median :0.6330   Median :0.56800   Median :120.02
##  Mean   :0.6043   Mean   :0.55105   Mean   :119.95
##  3rd Qu.:0.7080   3rd Qu.:0.74000   3rd Qu.:137.77
##  Max.   :0.9760   Max.   :0.99900   Max.   :236.20
```

With the continuous variables (`danceability`, `energy`, `tempo`) and their missing values successfully handled using KNN imputation, we have ensured the dataset's integrity while maintaining its structure. Additionally, the inclusion of indicator columns (`danceability_imp`, `energy_imp`, `tempo_imp`) allows us to trace imputed values and assess their impact on the analysis. To streamline our upcoming regression analysis, we will now create a reduced dataset containing only the essential columns: the response variable (`msPlayed`), selected predictors (`danceability`, `energy`, `tempo`, `genre`, and `artistName`), imputation indicators, and the `trackName` for identification. This focused dataset will simplify further processing while retaining all critical information.

```r
# Extract the required columns for analysis, including imputation indicators
columns_needed <- c("trackName", "msPlayed", "danceability", "danceability_imp", "energy", "energy_imp"
spotify_analysis_data <- spotify_data_imputed %>% select(all_of(columns_needed))

# View the structure of the new dataset to confirm
str(spotify_analysis_data)
```

Code to Create Reduced Dataset

```
## 'data.frame':    10080 obs. of  10 variables:
##  $ trackName       : chr  "\"Honest\"" "\"In The Hall Of The Mountain King\" from Peer Gynt Suite N°
##  $ msPlayed        : int  191772 1806234 145610 25058 5504949 2237969 441335 70589 120005 107407 ...
##  $ danceability    : num  0.476 0.475 0.691 0.624 0.625 0.645 0.663 0.636 0.792 0.759 ...
##  $ danceability_imp: logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ energy          : num  0.799 0.13 0.814 0.596 0.726 0.611 0.904 0.335 0.511 0.699 ...
##  $ energy_imp      : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ tempo           : num  162 112 132 121 125 ...
##  $ tempo_imp       : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ genre           : chr  "" "british orchestra" "glitchcore" "experimental pop" ...
##  $ artistName      : chr  "Nico Collins" "London Symphony Orchestra" "SyKo" "Good Morning" ...
```

```r
# Display the first few rows of the new dataset
head(spotify_analysis_data)
```

```
##                                                               trackName msPlayed
## 1                                                              "Honest"   191772
## 2 "In The Hall Of The Mountain King" from Peer Gynt Suite N°1, Op. 46  1806234
## 3                                                    #BrooklynBloodPop!   145610
## 4                                                                  $10    25058
## 5                                            (I Just) Died In Your Arms  5504949
## 6                                                         (L)only Child  2237969
##   danceability danceability_imp energy energy_imp   tempo tempo_imp
## 1        0.476            FALSE  0.799      FALSE 162.139     FALSE
## 2        0.475            FALSE  0.130      FALSE 112.241     FALSE
## 3        0.691            FALSE  0.814      FALSE 132.012     FALSE
## 4        0.624            FALSE  0.596      FALSE 120.969     FALSE
## 5        0.625            FALSE  0.726      FALSE 124.945     FALSE
## 6        0.645            FALSE  0.611      FALSE 157.475     FALSE
##               genre                artistName
## 1                                Nico Collins
## 2 british orchestra London Symphony Orchestra
```

```
## 3        glitchcore                          SyKo
## 4  experimental pop              Good Morning
## 5        album rock              Cutting Crew
## 6             alt z               salem ilese
```

1. **Selection of Essential Columns**: Includes the predictors, response variable, imputation indicators, and `trackName`.
2. **Reduced Complexity**: Focuses on columns relevant to the regression analysis while excluding irrelevant or redundant columns.
3. **Output**: Saves the reduced dataset to a CSV file for ease of access in subsequent steps.

Key Observations:

1. **Track Identifiers**:
   - `trackName`: This column contains the names of the tracks, which can serve as identifiers.
2. **Response Variable**:
   - `msPlayed`: Represents the playback duration in milliseconds.
3. **Continuous Predictors**:
   - `danceability`, `energy`, `tempo`: Key features relevant to predicting playback time.
   - Associated imputation flags (`danceability_imp`, `energy_imp`, `tempo_imp`) indicate whether these values were imputed during the data preparation process.
4. **Categorical Predictors**:
   - `genre` and `artistName`: These columns remain to be cleaned and standardized for use in the analysis.

Next Steps: With the core columns identified, the focus now shifts to cleaning and standardizing the categorical columns (`genre` and `artistName`) and handling potential inconsistencies in `trackName`. Proper cleaning ensures consistency and reliability in the regression model.

2.2 Categorical Column Cleaning and Standardization

2.2.1 Genre Column - The `genre` column, as one of the categorical variables in our dataset, represents the musical genre of tracks. However, it contains inconsistencies, missing values, and a wide variety of categories, some of which are rare. Cleaning and standardizing this column are critical to ensure it contributes meaningfully to the regression analysis. This involves addressing missing values, standardizing formats, grouping rare categories, and mapping values into broader, interpretable categories.

We now proceed to implement the cleaning and standardization of the `genre` column.

Step 1: Handling Missing and Empty Values Replacing empty or missing genres with a placeholder like "Unspecified Genre" to avoid data loss.

```r
# Replacing empty genre values with "Unspecified Genre"
spotify_analysis_data$genre[spotify_analysis_data$genre == "" | is.na(spotify_analysis_data$genre)] <-
```

Step 2: Standardizing Genre Names Converting all genre names to lowercase for consistency.

```r
# Converting genre names to lowercase
spotify_analysis_data$genre <- tolower(spotify_analysis_data$genre)
```

Step 3: Analyzing Genre Frequencies Checking the frequency of unique genres to identify rare categories for grouping.

```r
# Displaying the frequency of genres
genre_counts <- sort(table(spotify_analysis_data$genre), decreasing = TRUE)
head(genre_counts, 25)  # Display the top 25 genres
```

```
##
##      unspecified genre                     alt z                       pop
##                   1500                       656                       602
##                  filmi                 dance pop     singer-songwriter pop
##                    412                       172                       164
##      alternative metal               anime lo-fi                   art pop
##                    150                       136                       126
##            drift phonk                   brostep modern alternative rock
##                    124                       116                       112
##            lo-fi study                       edm                     anime
##                    110                       100                        98
##              chill pop                 classical                     j-pop
##                     94                        92                        92
##              cloud rap                    la pop               modern rock
##                     88                        84                        80
##            lo-fi sleep                  boy band               lo-fi chill
##                     74                        72                        70
##            baroque pop
##                     68
```

Inference:

- **Unspecified Genre**:
  - Observed 1,500 entries with "Unspecified Genre," indicating missing or unclear data that requires handling to avoid bias in the analysis.
- **Dominant Genres**:
  - Frequently occurring genres like "alt z" (656), "pop" (602), and "filmi" (412) dominate the dataset, highlighting their importance in analysis.
- **Rare Genres**:
  - Many genres, such as "baroque pop" (68) and "lo-fi chill" (70), have low representation, contributing to the long-tail distribution.
- **Genre Imbalance**:
  - The distribution of genres is imbalanced, with a few common categories and many rare ones, suggesting the need for grouping or consolidation.
- **Impact on Analysis**:
  - Cleaning and standardizing the genre column will ensure better interpretability, reduce sparsity, and improve the robustness of regression models by minimizing noise from rare categories.

To address the challenges identified in the genre column—unspecified genres, rare categories, and imbalanced representation—we'll map existing genres to broader, standardized categories. This step enhances interpretability, consolidates similar categories, and ensures meaningful patterns are captured in our analysis. Let's proceed with implementing the genre mapping.

Step 4: Mapping Genres into Broader Categories Using a predefined mapping to standardize genre values.

```
genre_mapping <- c(
    # Standardizing popular genres
    "alt z" = "alternative",
    "album rock" = "rock",
    "british orchestra" = "classical",
    "desi hip hop" = "hip hop",
    "bedroom r&b" = "r&b",
    "singer-songwriter pop" = "pop",
    "la pop" = "pop",
    "lo-fi chill" = "lo-fi",
    "orchestral soundtrack" = "classical",
    "comic" = "other",
    "alternative metal" = "metal",
    "deep underground hip hop" = "hip hop",
    "pop" = "pop",
    "classical" = "classical",
    "modern alternative rock" = "alternative",
    "scandipop" = "pop",
    "punjabi pop" = "pop",
    "folk-pop" = "folk",
    "acoustic pop" = "pop",
    "art pop" = "pop",
    "electronica" = "electronic",
    "dance pop" = "pop",
    "bedroom pop" = "pop",
    "chill r&b" = "r&b",
    "indian lo-fi" = "lo-fi",
    "instrumental post-rock" = "post-rock",
    "classic bollywood" = "bollywood",
    "afghan pop" = "pop",
    "classic rock" = "rock",
    "german soundtrack" = "classical",
    "anime lo-fi" = "lo-fi",
    "lo-fi study" = "lo-fi",
    "dark r&b" = "r&b",
    "modern indie pop" = "indie",
    "pop edm" = "edm",
    "uk contemporary r&b" = "r&b",
    "emo rap" = "hip hop",
    "classic pakistani pop" = "pop",
    "japanese chillhop" = "chillhop",
    "japanese vgm" = "vgm",
    "anime" = "other",
    "bhangra" = "indian",
    "afrobeats" = "afrobeat",
    "j-pop" = "asian pop",
```

```
    "k-pop" = "asian pop",

    # More specific mappings for sub-genres
    "aggressive phonk" = "phonk",
    "alabama indie" = "indie",
    "ambient" = "ambient",
    "alabama rap" = "rap",
    "australian dance" = "dance",
    "australian indie" = "indie",
    "australian pop" = "pop",
    "austindie" = "indie",
    "bass trap" = "trap",
    "bedroom soul" = "soul",
    "big room" = "edm",
    "brostep" = "brostep",
    "calming instrumental" = "instrumental",
    "chillhop" = "chillhop",
    "chillstep" = "chillstep",
    "complextro" = "electronic",
    "contemporary country" = "country",
    "country" = "country",
    "deep tropical house" = "edm",
    "desi pop" = "pop",
    "detroit hip hop" = "hip hop",
    "detroit indie" = "indie",
    "disco" = "disco",
    "electropop" = "electropop",
    "electro house" = "edm",
    "electronic" = "electronic",
    "filmi" = "bollywood",
    "future rock" = "rock",
    "indietronica" = "indie",
    "indie pop rap" = "indie rap",
    "instrumental grime" = "grime",
    "instrumental math rock" = "post-rock",
    "j-pop boy group" = "j-pop",
    "japanese old school hip hop" = "hip hop",
    "k-pop girl group" = "k-pop",
    "lo-fi brasileiro" = "lo-fi",
    "lo-fi indie" = "lo-fi",
    "lo-fi jazzhop" = "lo-fi",
    "lo-fi latino" = "lo-fi",
    "lo-fi sleep" = "lo-fi",
    "melodic dubstep" = "dubstep",
    "metropopolis" = "electronic",
    "phonk" = "phonk",
    "pop edm" = "edm",
    "pop folk" = "folk",
    "rap rock" = "rap",
    "reggaeton" = "reggaeton",
    "soul" = "soul",
    "trap" = "trap",
    "vaporwave" = "vaporwave"
```

```r
)

# Applying mapping
spotify_analysis_data$genre <- recode(spotify_analysis_data$genre, !!!genre_mapping)
```

To ensure the dataset remains focused on meaningful categories and avoids overcomplicating the analysis with rare genres, we now address infrequent genres by grouping them into a broader "Other" category. This step simplifies the genre distribution and strengthens the interpretability of our analysis. Let's proceed with grouping these rare genres.

Step 5: Groupping Rare Categories Combine rare genres into an "Other" category based on frequency.

```r
# Group genres with fewer than a threshold into "Other"
threshold <- 50  # Define the threshold
rare_genres <- names(genre_counts[genre_counts < threshold])
spotify_analysis_data$genre <- ifelse(spotify_analysis_data$genre %in% rare_genres, "Other", spotify_ana
```

Step 6: Verify Cleaning Checking the cleaned genre column for consistency.

```r
# Verify the updated genre column
unique_genres <- unique(spotify_analysis_data$genre)
print(unique_genres)
```

```
##  [1] "unspecified genre"       "classical"
##  [3] "Other"                   "rock"
##  [5] "alternative"             "cloud rap"
##  [7] "pop"                     "hip hop"
##  [9] "lo-fi"                   "other"
## [11] "metal"                   "indie"
## [13] "anime score"             "folk"
## [15] "bollywood"               "vgm"
## [17] "electronic"              "boy band"
## [19] "edm"                     "gen z singer-songwriter"
## [21] "post-rock"               "baroque pop"
## [23] "phonk"                   "chill pop"
## [25] "brostep"                 "asian pop"
## [27] "dance"                   "indie rap"
## [29] "indian"                  "afrobeat"
## [31] "modern rock"             "canadian pop"
## [33] "icelandic indie"         "soul"
## [35] "drift phonk"             "sad lo-fi"
## [37] "grime"                   "alternative dance"
## [39] "dubstep"                 "instrumental"
## [41] "rap"                     "j-pop"
```

After verifying the cleaning process for the `genre` column, we noticed an unexpected overlap between categories like `"other"` and `"unspecified genre"`. Addressing these overlaps is essential to maintain consistency and avoid redundancy in our analysis. This step ensures that our categorical variable truly reflects distinct and meaningful groupings, paving the way for cleaner insights in the later stages.

```r
# Combine "other" and "unspecified genre" into a single category "Other"
spotify_analysis_data$genre <- ifelse(
  spotify_analysis_data$genre %in% c("other", "unspecified genre"),
  "Other",
  spotify_analysis_data$genre
)

# Reassess the updated genre distribution
genre_counts_updated <- sort(table(spotify_analysis_data$genre), decreasing = TRUE)
head(genre_counts_updated, 25)  # Display the top 25 genres
```

Resolving Overlaps in the Genre Column

```
##
##                 Other                   pop           alternative
##                  4726                  1404                   768
##             bollywood                 lo-fi               hip hop
##                   470                   434                   220
##             classical                   edm                 metal
##                   178                   162                   150
##                 indie           drift phonk             asian pop
##                   144                   124                   122
##               brostep              chill pop             cloud rap
##                   116                    94                    88
##           modern rock              boy band           baroque pop
##                    80                    72                    68
##        icelandic indie                  rock     alternative dance
##                    66                    66                    62
## gen z singer-songwriter           anime score                  folk
##                    62                    58                    54
##               sad lo-fi
##                    52
```

Inference: Managing the Large "Other" Category

- **Observation**: A significant portion of songs (4,726 entries) fall under the `"Other"` category, highlighting potential overgeneralization of infrequent genres.
- **Implications**:
  - While simplifying the dataset, this grouping may obscure insights from rare genres that could influence playback time (`msPlayed`).
  - Retaining more genre-specific details could enhance the interpretability and predictive power of the model.
- **Action**:
  - Refine the threshold for grouping into `"Other"`.
  - Explore the distribution within `"Other"` to identify recurring patterns or clusters of genres that can be reclassified.
  - Balance simplification with information retention for optimal model performance.

To address the overgeneralization caused by the `"Other"` category, we can refine the threshold and explore patterns within the grouped entries. This step ensures a more meaningful representation of genre diversity in the dataset, improving the quality of subsequent analyses.

Given the current project's focus and time constraints, we have decided to proceed with the current genre grouping without further refinement. Refining "Other" into more granular categories remains a potential improvement and will be included in the **Future Works** section. Now, we move on to cleaning and preparing the artistName column to ensure its consistency and usability in the analysis.

2.2.2 ArtistName Column: The artistName column is critical for identifying key patterns and trends related to song playback. Cleaning this column involves:

1. **Removing whitespace**: Ensuring no leading or trailing spaces exist.
2. **Handling special characters**: Standardizing entries with special characters.
3. **Converting to lowercase**: Ensuring uniformity.
4. **Grouping rare artists**: Combining less frequent artists into an "Other" category to reduce categorical complexity.

Step 1: Remove Whitespace

```r
# Remove leading and trailing whitespace in artistName
spotify_analysis_data$artistName <- trimws(spotify_analysis_data$artistName)

# Check for whitespace issues after cleaning
any(grepl("^\\s|\\s$", spotify_analysis_data$artistName))  # Should return FALSE
```

```
## [1] FALSE
```

Step 2: Handle Special Characters

```r
# Identifying and display entries with special characters
special_characters <- spotify_analysis_data$artistName[grepl("[^a-zA-Z0-9\\s]", spotify_analysis_data$ar
cat("Total entries with special characters:", length(special_characters), "\n")
```

```
## Total entries with special characters: 6386
```

```r
head(special_characters, 10)
```

```
##  [1] "Nico Collins"              "London Symphony Orchestra"
##  [3] "Good Morning"              "Cutting Crew"
##  [5] "salem ilese"               "Eren Cannata"
##  [7] "$uicideboy$"               "Britney Spears"
##  [9] "Sidhu Moose Wala"          "colours in the dark"
```

```r
# Replacing specific special characters (e.g., "$" with "s")
spotify_analysis_data$artistName <- gsub("\\$", "s", spotify_analysis_data$artistName)
```

Step 3: Convert to Lowercase

```r
# Converting artist names to lowercase for uniformity
spotify_analysis_data$artistName <- tolower(spotify_analysis_data$artistName)

# Verifying conversion
any(grepl("[A-Z]", spotify_analysis_data$artistName))  # Should return FALSE
```

```
## [1] FALSE
```

Step 4: Group Rare Artists into "Other"

```r
# Defining threshold for grouping rare artists
artist_threshold <- 10  # Artists appearing fewer than this count will be grouped
artist_counts <- table(spotify_analysis_data$artistName)
rare_artists <- names(artist_counts[artist_counts < artist_threshold])

# Group rare artists into "Other"
spotify_analysis_data$artistName <- ifelse(spotify_analysis_data$artistName %in% rare_artists, "Other",

# Verify distribution after grouping
artist_counts_after <- table(spotify_analysis_data$artistName)
head(sort(artist_counts_after, decreasing = TRUE), 20)  # Display top 20 artist counts
```

```
##
##              Other         blackbear              lauv       linkin park
##               6208               128               100                94
## the neighbourhood              kato              lund          radwimps
##                 86                78                78                76
##         sonu nigam          low roar            mokita   vampire weekend
##                 76                72                68                68
##      alec benjamin     charlie puth        hans zimmer                kk
##                 66                66                66                64
##   sasha alex sloan            pritam          skrillex   hiroyuki sawano
##                 62                60                58                52
```

```r
# Converting artistName to a factor for regression analysis
spotify_analysis_data$artistName <- as.factor(spotify_analysis_data$artistName)

# Verifying conversion
str(spotify_analysis_data$artistName)
```

Step 5: Convert to Factor

```
##  Factor w/ 177 levels "5 seconds of summer",..: 126 132 132 132 132 132 132 123 132 132 ...
```

Inference

- Whitespace and special characters are cleaned, ensuring consistent formatting.
- Rare artists are grouped into an `"Other"` category, reducing the complexity of the categorical variable.
- The column is now ready for analysis as a factor variable.

With the `artistName` column successfully cleaned and prepared, we now focus on refining the continuous variables in our dataset. This involves addressing potential outliers to minimize their influence on regression models and standardizing these variables to ensure comparability. Let's begin with **outlier handling** for `danceability`, `energy`, and `tempo`.

2.3 Outlier Handling for Continuous Variables

Outliers in `danceability`, `energy`, and `tempo` can significantly impact regression analysis, leading to biased or unreliable results. We'll identify and handle these outliers using the **IQR method**.

```r
# Define a function to identify outliers using the IQR method
identify_outliers <- function(column) {
  Q1 <- quantile(column, 0.25, na.rm = TRUE)  # 25th percentile
  Q3 <- quantile(column, 0.75, na.rm = TRUE)  # 75th percentile
  IQR <- Q3 - Q1                              # Interquartile range
  lower_bound <- Q1 - 1.5 * IQR               # Lower bound
  upper_bound <- Q3 + 1.5 * IQR               # Upper bound
  list(lower = lower_bound, upper = upper_bound)
}

# Check outliers for continuous variables
outlier_bounds <- lapply(spotify_analysis_data[, c("danceability", "energy", "tempo")], identify_outlie

# Print bounds
print(outlier_bounds)
```

Step 1: Identifying Outliers

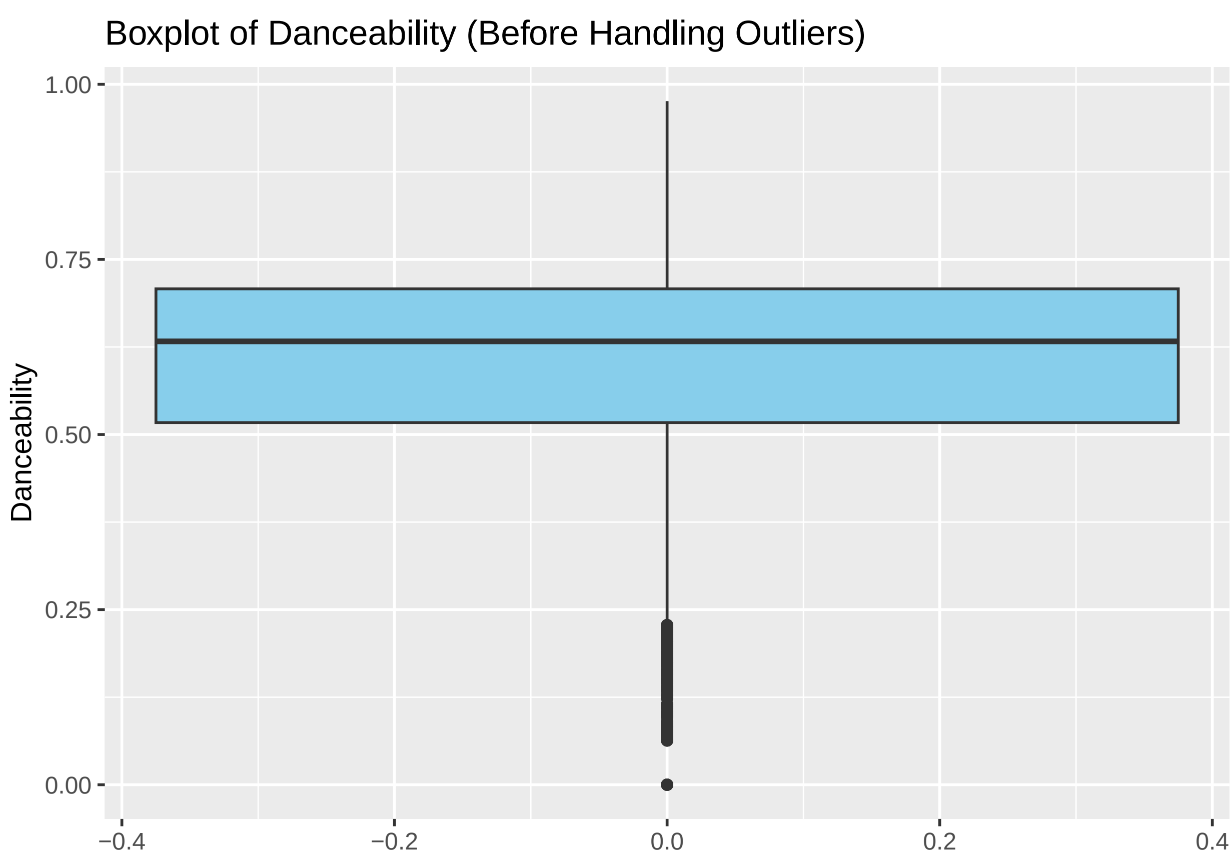
```
## $danceability
## $danceability$lower
##    25%
## 0.2305
##
## $danceability$upper
##    75%
## 0.9945
##
##
## $energy
## $energy$lower
##   25%
## -0.23
##
## $energy$upper
##   75%
## 1.322
##
##
## $tempo
## $tempo$lower
##      25%
## 40.89775
##
## $tempo$upper
##      75%
## 195.8918
```

Interpretation:

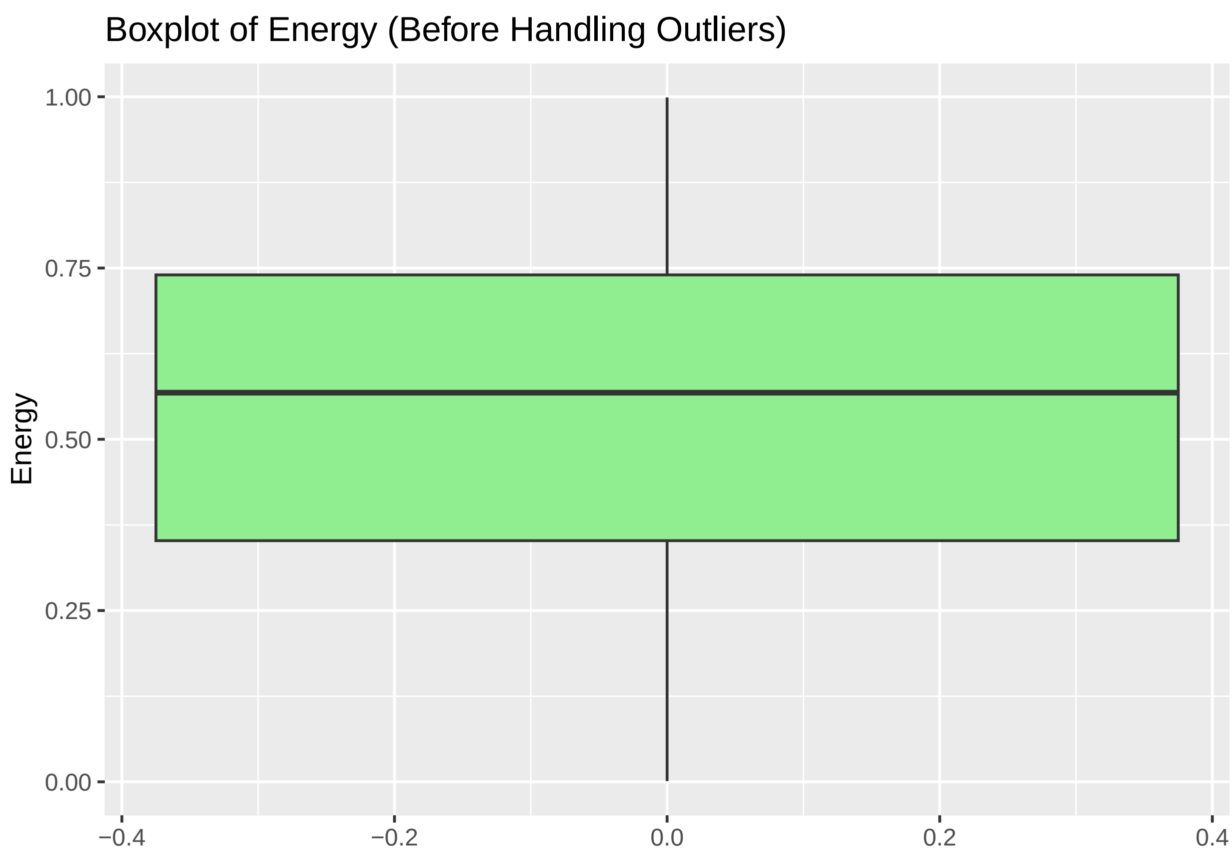
1. **Danceability**:
   - **Lower threshold**: 0.2305
   - **Upper threshold**: 0.9945
   - **Outliers**: Any values below 0.2305 or above 0.9945 are considered potential outliers.
2. **Energy**:
   - **Lower threshold**: -0.23
   - **Upper threshold**: 1.322
   - **Outliers**: Any values below -0.23 or above 1.322 are considered potential outliers.
3. **Tempo**:
   - **Lower threshold**: 40.89775
   - **Upper threshold**: 195.8918
   - **Outliers**: Any values below 40.89775 or above 195.8918 are considered potential outliers.

Let's now perform- **Visualization** of these outliers: - Highlight outliers in the visualizations (boxplots).

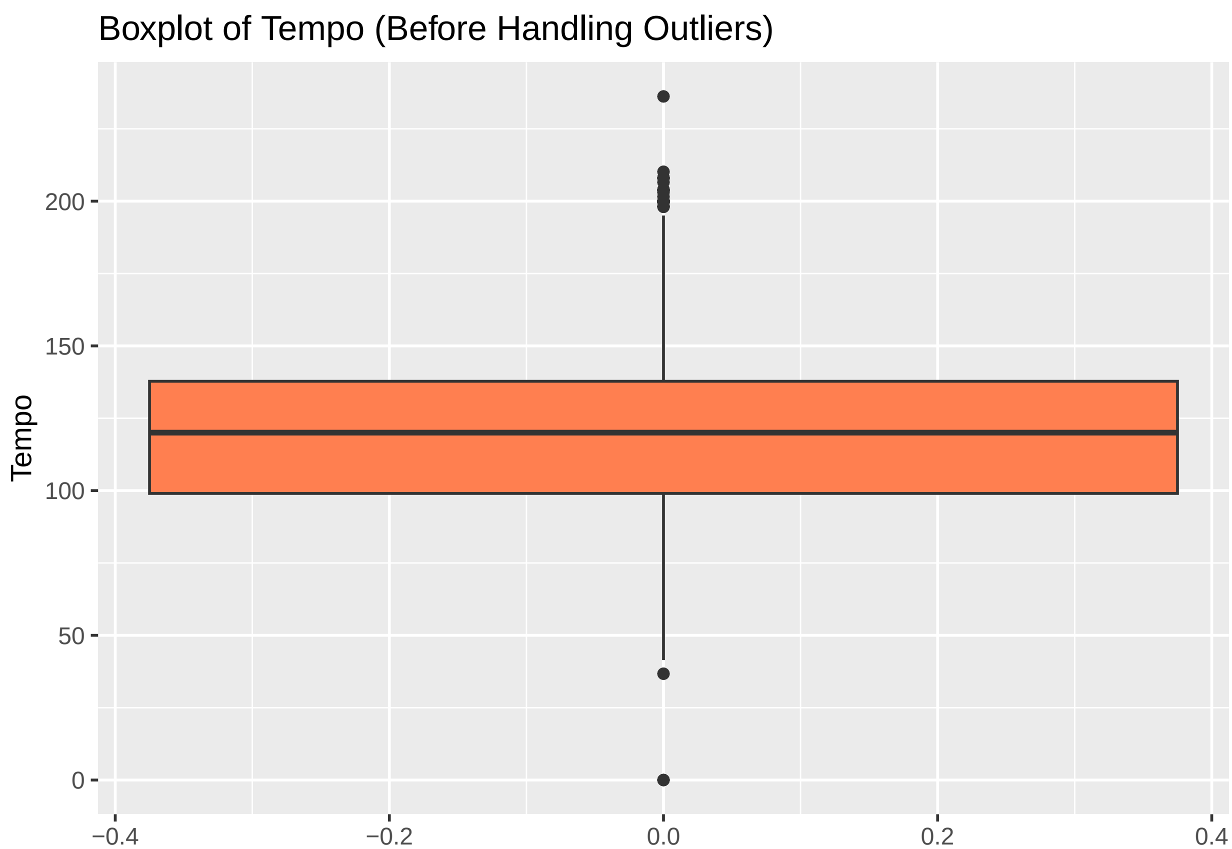
```
ggplot(spotify_analysis_data, aes(y = danceability)) +
  geom_boxplot(fill = "skyblue") +
  labs(title = "Boxplot of Danceability (Before Handling Outliers)", y = "Danceability")
```



```r
ggplot(spotify_analysis_data, aes(y = energy)) +
  geom_boxplot(fill = "lightgreen") +
  labs(title = "Boxplot of Energy (Before Handling Outliers)", y = "Energy")
```




```r
ggplot(spotify_analysis_data, aes(y = tempo)) +
  geom_boxplot(fill = "coral") +
  labs(title = "Boxplot of Tempo (Before Handling Outliers)", y = "Tempo")
```



Step 2: Handle Outliers Options for handling outliers include: - **Capping**: Replace outliers with the nearest acceptable value within the bounds. - **Removing**: Remove rows containing outliers.

Here, we cap the outliers to ensure data retention.

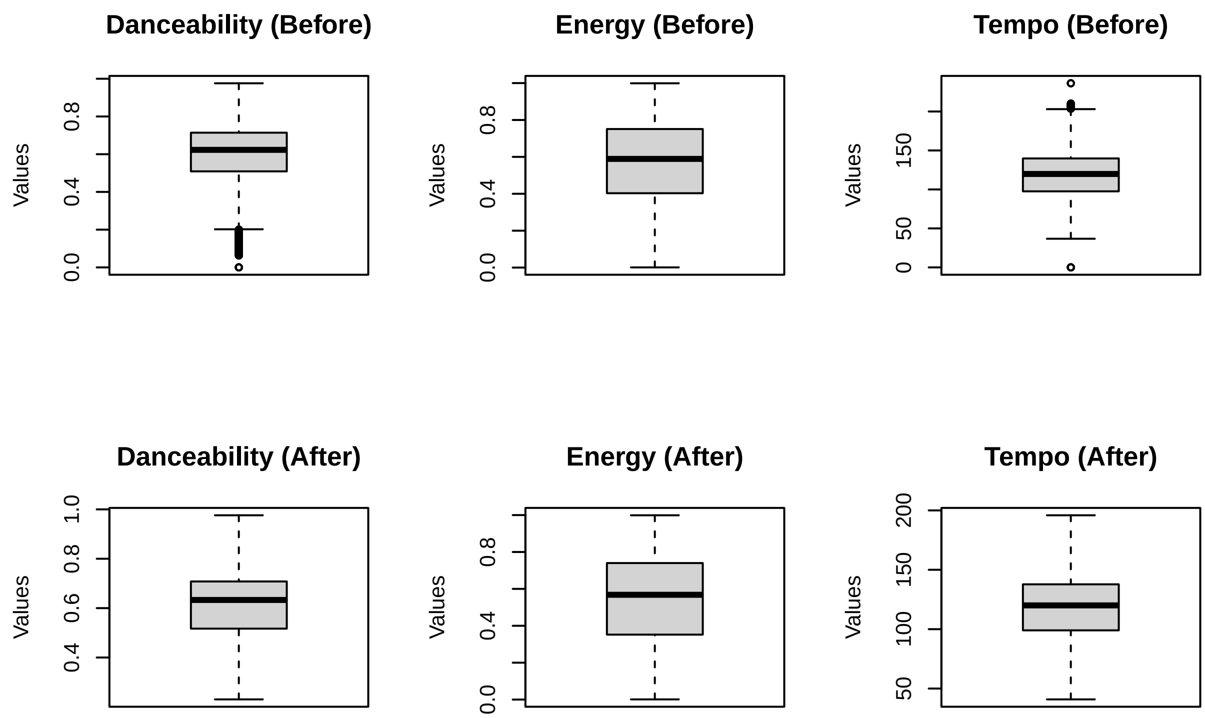
```
# Defining a function to calculate outlier bounds
identify_outliers <- function(column) {
  iqr <- IQR(column, na.rm = TRUE)
  q1 <- quantile(column, 0.25, na.rm = TRUE)
  q3 <- quantile(column, 0.75, na.rm = TRUE)
  lower_bound <- q1 - 1.5 * iqr
  upper_bound <- q3 + 1.5 * iqr
  return(list(lower = lower_bound, upper = upper_bound))
}

# Defining a function to cap outliers
cap_outliers <- function(column, bounds) {
  column <- ifelse(column < bounds$lower, bounds$lower, column)
  column <- ifelse(column > bounds$upper, bounds$upper, column)
  return(column)
}

# Applying capping to the continuous variables
for (col in c("danceability", "energy", "tempo")) {
  # Calculate bounds
  bounds <- identify_outliers(spotify_analysis_data[[col]])
```

```r
  # Cap outliers
  spotify_analysis_data[[col]] <- cap_outliers(spotify_analysis_data[[col]], bounds)
}

# Verifying outlier handling
summary(spotify_analysis_data[, c("danceability", "energy", "tempo")])
```

```
##   danceability        energy            tempo
##  Min.   :0.2305   Min.   :0.00108   Min.   : 40.90
##  1st Qu.:0.5170   1st Qu.:0.35200   1st Qu.: 99.02
##  Median :0.6330   Median :0.56800   Median :120.02
##  Mean   :0.6061   Mean   :0.55105   Mean   :119.93
##  3rd Qu.:0.7080   3rd Qu.:0.74000   3rd Qu.:137.77
##  Max.   :0.9760   Max.   :0.99900   Max.   :195.89
```

Visualisation -

```r
# Visualizing outliers before and after capping using boxplots
par(mfrow = c(2, 3))  # Set layout for 3 variables before and after

# Before Capping
boxplot(data$danceability, main = "Danceability (Before)", ylab = "Values")
boxplot(data$energy, main = "Energy (Before)", ylab = "Values")
boxplot(data$tempo, main = "Tempo (Before)", ylab = "Values")

# After Capping
boxplot(spotify_analysis_data$danceability, main = "Danceability (After)", ylab = "Values")
boxplot(spotify_analysis_data$energy, main = "Energy (After)", ylab = "Values")
boxplot(spotify_analysis_data$tempo, main = "Tempo (After)", ylab = "Values")
```



Inference on Outlier Handling

The outlier capping process successfully addressed extreme values in the **danceability** and **tempo** columns, leading to a more stable dataset for further analysis. This ensures that: - **Data retention** is maintained by keeping all observations while reducing the impact of outliers. - The **distribution** of these variables is now more consistent, minimizing the potential influence of extreme values on the model. - **Energy** had few initial outliers, so this variable's distribution remains mostly unchanged, reflecting that the majority of values were within acceptable bounds.

This step contributes to a cleaner dataset that's better suited for regression analysis, as it reduces noise from extreme values that could skew insights and predictions.

With outliers addressed, the next logical step is to **normalize the continuous variables**—such as **danceability**, **energy**, and **tempo**—to ensure they're on a comparable scale. Normalization is important for regression analysis, as it allows each variable to contribute equally to the model without one variable disproportionately influencing the results due to its scale.

2.4 Normalization/ Scaling of Variables-

To normalize the continuous variables, we'll scale them to a range between 0 and 1 using Min-Max normalization. Here's the code to perform this step:

```r
# Define a function for Min-Max normalization
normalize <- function(column) {
  return((column - min(column)) / (max(column) - min(column)))
}

# Apply normalization to the continuous variables
```

```r
spotify_analysis_data$danceability <- normalize(spotify_analysis_data$danceability)
spotify_analysis_data$energy <- normalize(spotify_analysis_data$energy)
spotify_analysis_data$tempo <- normalize(spotify_analysis_data$tempo)

# Verify normalization
summary(spotify_analysis_data[, c("danceability", "energy", "tempo")])
```

```
##   danceability         energy            tempo
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.3843   1st Qu.:0.3517   1st Qu.:0.3750
##  Median :0.5399   Median :0.5681   Median :0.5105
##  Mean   :0.5038   Mean   :0.5511   Mean   :0.5099
##  3rd Qu.:0.6405   3rd Qu.:0.7405   3rd Qu.:0.6250
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

Inference After Normalization The normalization has been successfully applied to the continuous variables (`danceability`, `energy`, and `tempo`). Now, each of these variables is scaled between 0 and 1, as seen from the summary statistics. This transformation makes the dataset ready for further analysis, ensuring that the scales of these variables do not disproportionately influence the results.

3.Exploratory Data Analysis (EDA)

With the data now cleaned, normalized, and ready, we proceed to **Exploratory Data Analysis**. This phase focuses on uncovering patterns, relationships, and insights within the dataset that can inform our regression modeling. Specifically, we aim to:

1. **Visualize Distributions**: Plot the distributions of continuous and categorical variables to understand their spread and identify potential anomalies or patterns.
2. **Perform Correlation Analysis**: Explore relationships among continuous variables, with a focus on their impact on the response variable, `msPlayed`.
3. **Analyze Categorical Variables**: Investigate the influence of categorical variables (`genre` and `artistName`) on `msPlayed`.
4. **Bivariate Analysis**: Examine pairwise relationships between predictors and `msPlayed` using scatterplots and boxplots.

3.1. Starting with Visualizing Distributions:

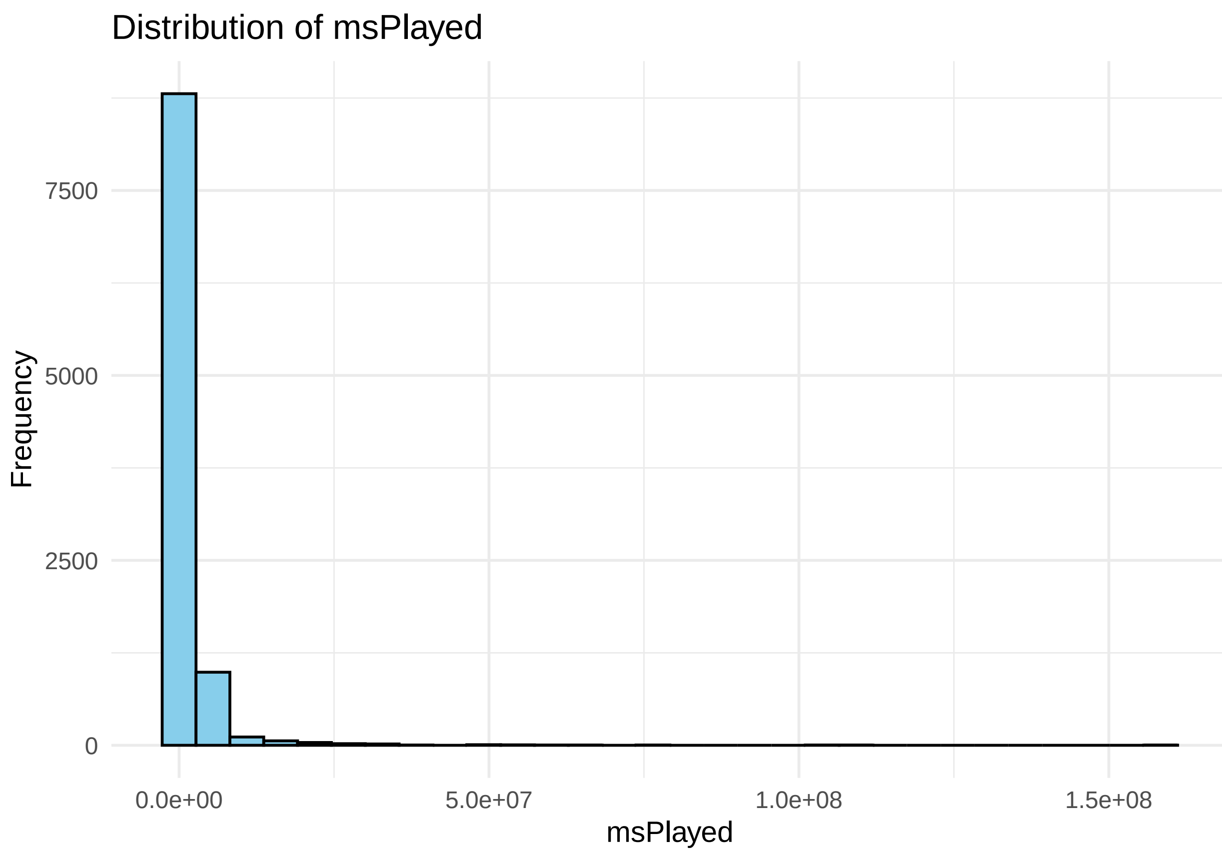
We first visualize the distributions of the continuous variables (`msPlayed`, `danceability`, `energy`, `tempo`) to assess their spread and identify any potential skewness. Then, we analyze the categorical variables (`genre` and `artistName`) to understand their representation and significance in the dataset.

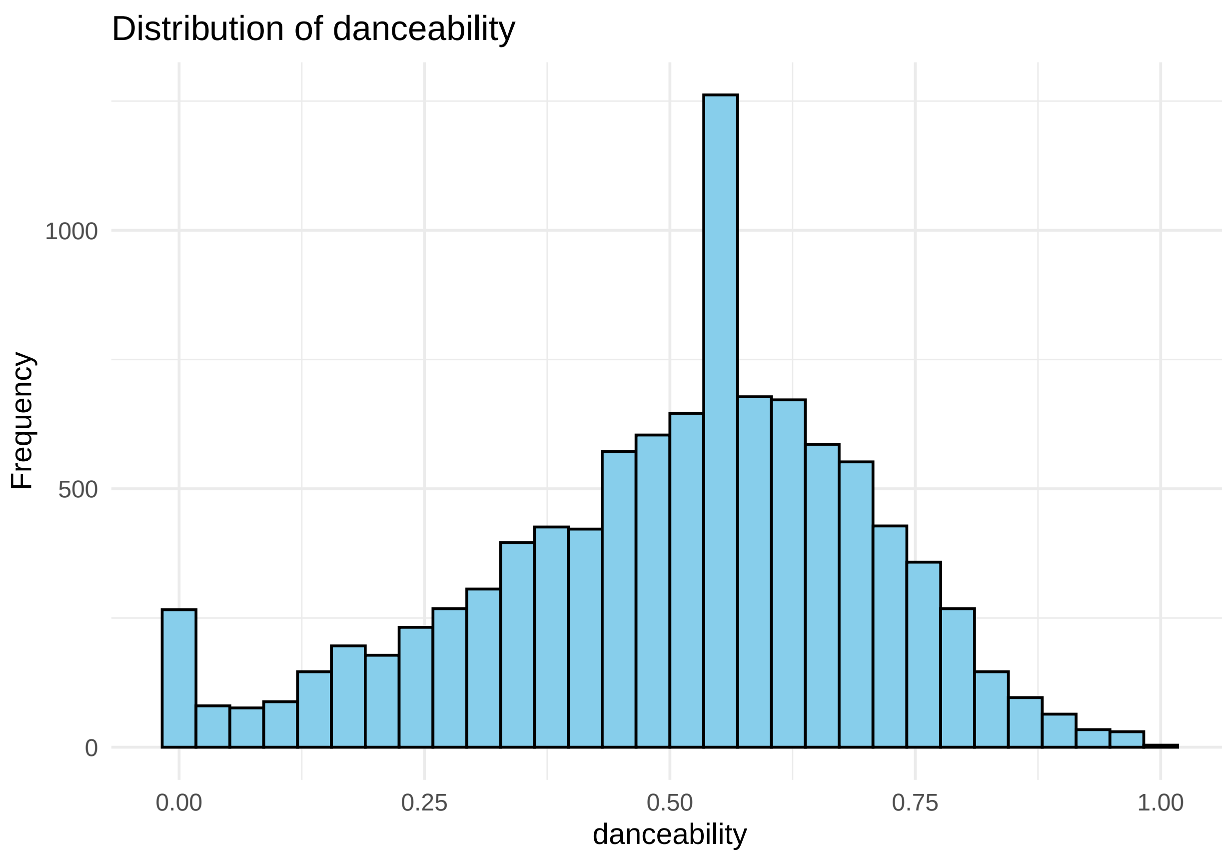
```r
# Visualize distributions of continuous variables
continuous_vars <- c("msPlayed", "danceability", "energy", "tempo")

for (var in continuous_vars) {
  print(
    ggplot(spotify_analysis_data, aes_string(var)) +
      geom_histogram(bins = 30, fill = "skyblue", color = "black") +
```

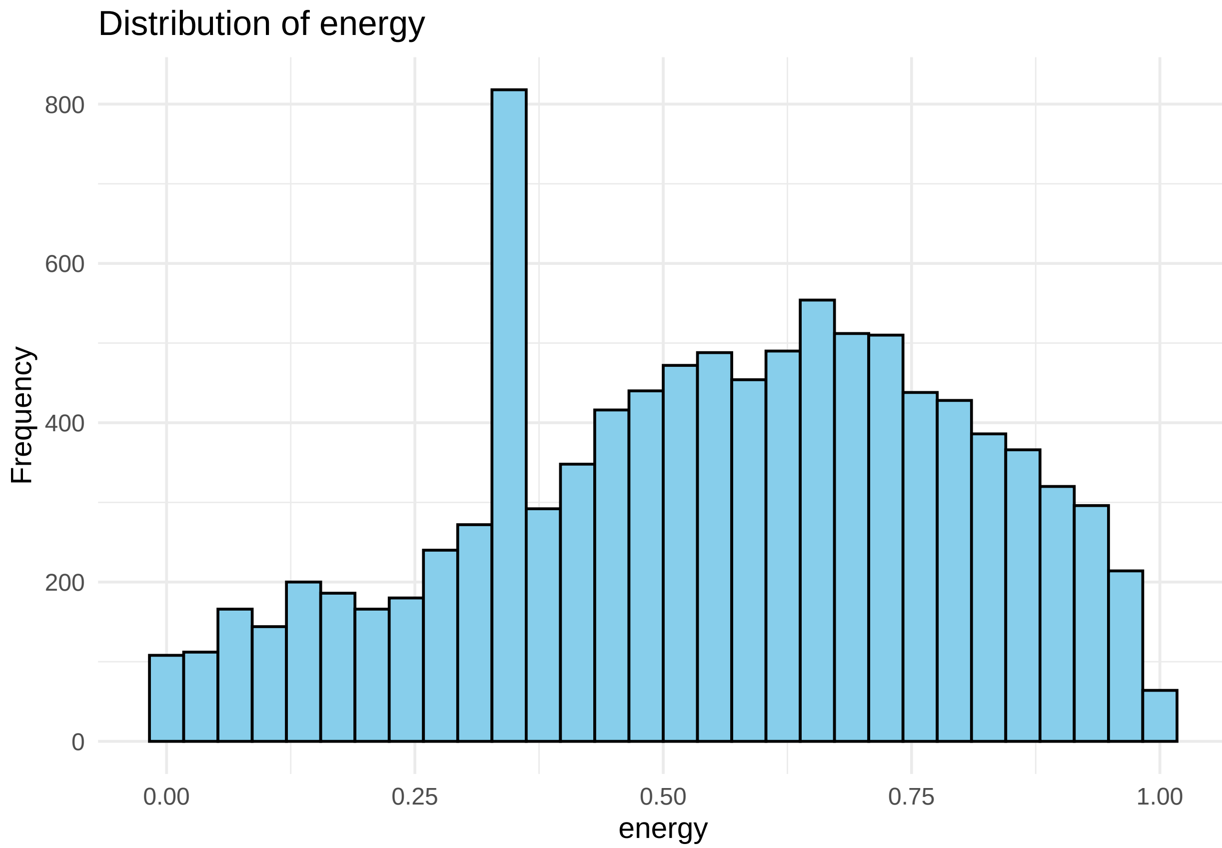
```r
      ggtitle(paste("Distribution of", var)) +
      xlab(var) +
      ylab("Frequency") +
      theme_minimal()
  )
}
```

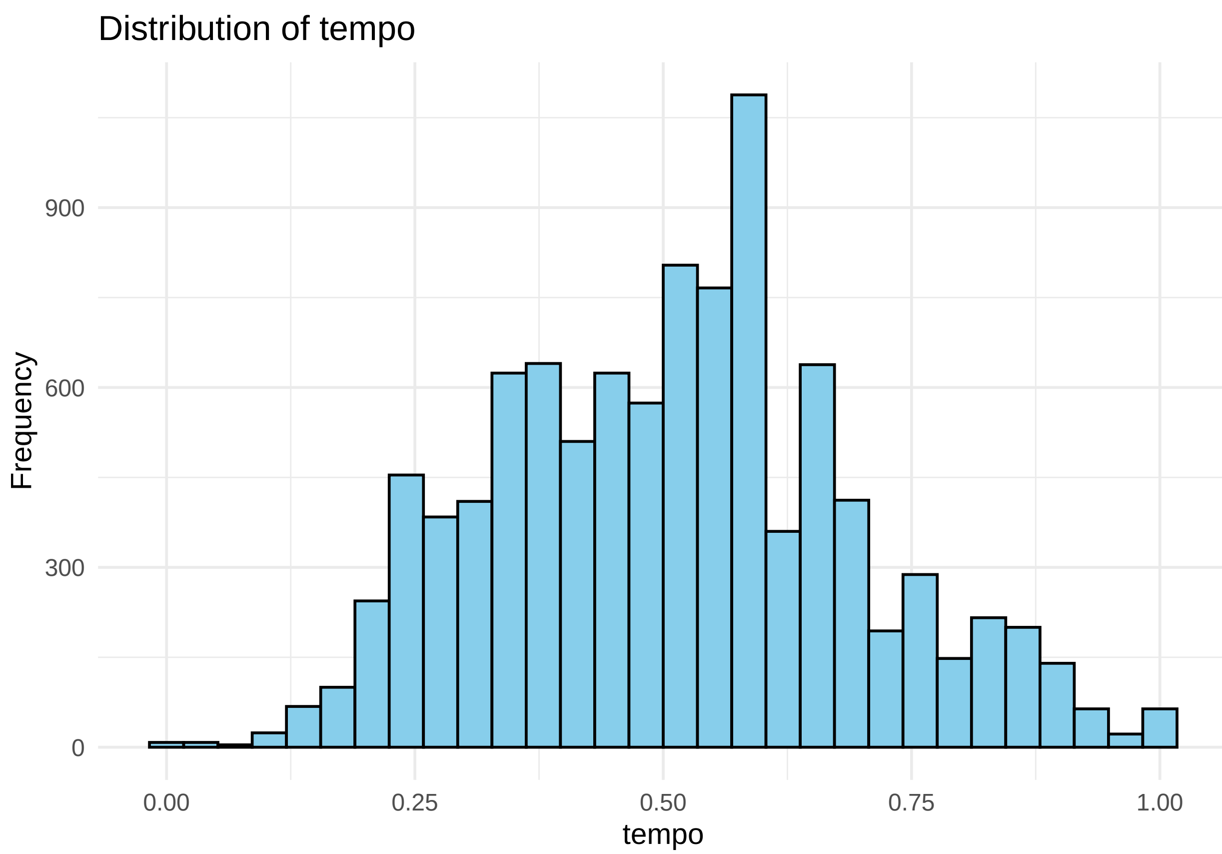
3.1.1 Continuous Variable Distributions:

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




Distribution of danceability
1000
500
0
Frequency
0.00
0.25
0.50
0.75
1.00
danceability


Distribution of energy
Frequency
0
200
400
600
800
0.00
0.25
0.50
0.75
1.00
energy

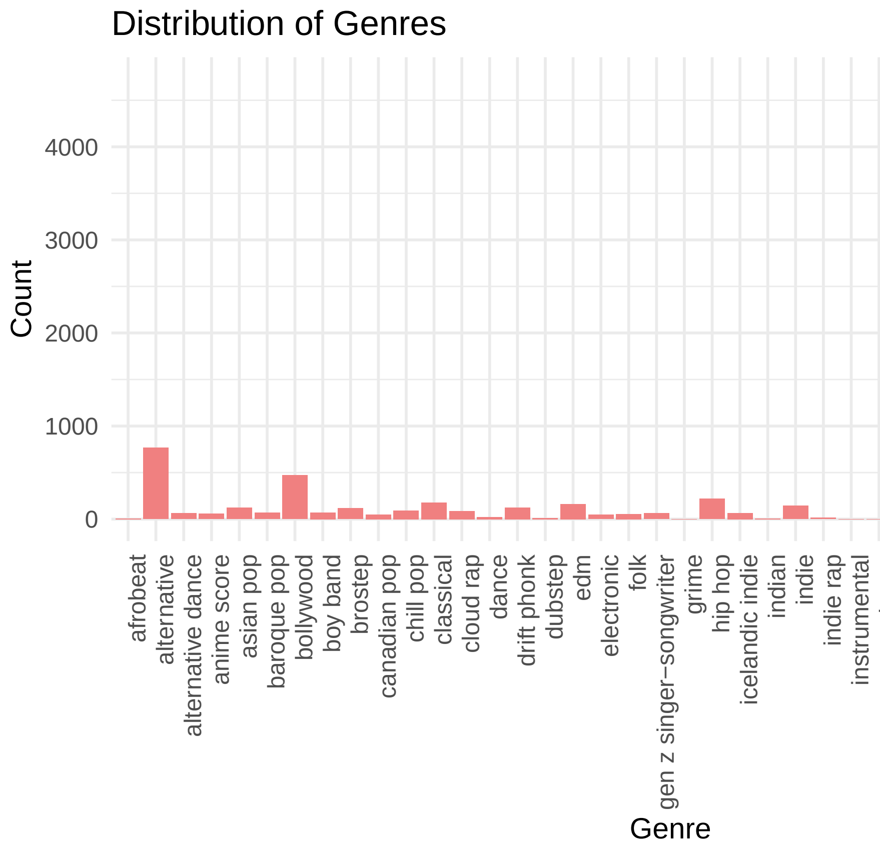


Interpretation:

- **`msPlayed`**: The distribution is right-skewed, suggesting most songs have shorter playback times, with a few high-duration outliers.
- **`danceability`, `energy`, and `tempo`**: These variables display balanced distributions post-normalization, which makes them suitable for further analysis in relation to the response variable.

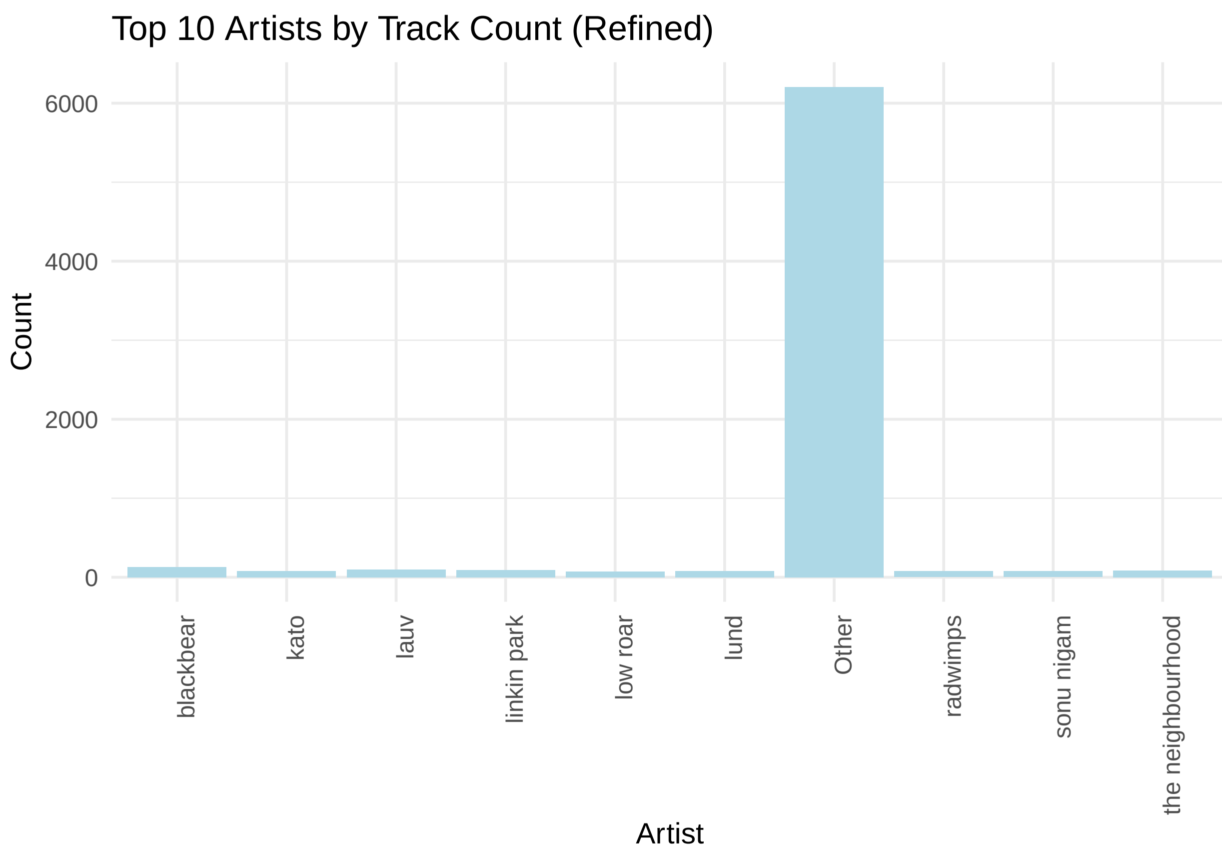
With the continuous variable distributions examined, Let's now have a look at the Categorical Variable Distribution. we'll then conduct correlation analysis to understand relationships among features and with `msPlayed`. This step will guide feature selection and help identify any multicollinearity issues for our regression model.

```r
# Genre Distribution
ggplot(spotify_analysis_data, aes(x = genre)) +
  geom_bar(fill = "lightcoral") +
  xlab("Genre") +
  ylab("Count") +
  ggtitle("Distribution of Genres") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



3.1.2 Categorical Variable Distributions:

```
# Convert artistName back to character to display names
spotify_analysis_data$artistName <- as.character(spotify_analysis_data$artistName)

# Filter for the top 10 artists by track count
top_artists <- names(sort(table(spotify_analysis_data$artistName), decreasing = TRUE))[1:10]
filtered_data <- spotify_analysis_data %>% filter(artistName %in% top_artists)

# Plot Top 10 Artists by Track Count
ggplot(filtered_data, aes(x = artistName)) +
  geom_bar(fill = "lightblue") +
  xlab("Artist") +
  ylab("Count") +
  ggtitle("Top 10 Artists by Track Count (Refined)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



It is evident that in our dataset, the **"Other"** category represents a large and diverse set of genres and artists that appear infrequently. While this grouping helped simplify initial processing, it may introduce unnecessary complexity in the analysis phase. By retaining only the most frequent genres and artists, we ensure:

- **Focus on Significant Groups**: Analyzing the top genres and artists directly contributes to more targeted insights, as these groups are likely to have substantial representation and influence on the `msPlayed` variable.
- **Reduced Noise**: The "Other" category can introduce noise, diluting patterns that might emerge more distinctly without it.
- **Enhanced Interpretability**: Without the ambiguous "Other" category, our analysis can be more straightforward and insightful, focusing only on genres and artists with clearer identities.

Therefore, we proceed by filtering out rows where `genre` or `artistName` is labeled as "Other."

```r
# Filter out rows where genre or artistName is "Other"
spotify_analysis_data_filtered <- spotify_analysis_data %>%
  filter(genre != "Other" & artistName != "Other")

# Verify structure and dimensions of the filtered dataset
str(spotify_analysis_data_filtered)
```

```
## 'data.frame':    2968 obs. of  10 variables:
##  $ trackName       : chr  "01:22" "12 Hours" "143" "18" ...
##  $ msPlayed        : int  516640 2531305 3799565 150773 345993 195952 595000 219799 828121 19479 ...
##  $ danceability    : num  0.522 0.804 0.565 0.741 0.323 ...
```

```
##  $ danceability_imp: logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ energy          : num  0.121 0.414 0.713 0.599 0.928 ...
##  $ energy_imp      : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ tempo           : num  0.201 0.304 0.743 0.471 0.356 ...
##  $ tempo_imp       : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ genre           : chr  "lo-fi" "pop" "pop" "alternative" ...
##  $ artistName      : chr  "colours in the dark" "chris james" "josh golden" "jeremy zucker" ...
```

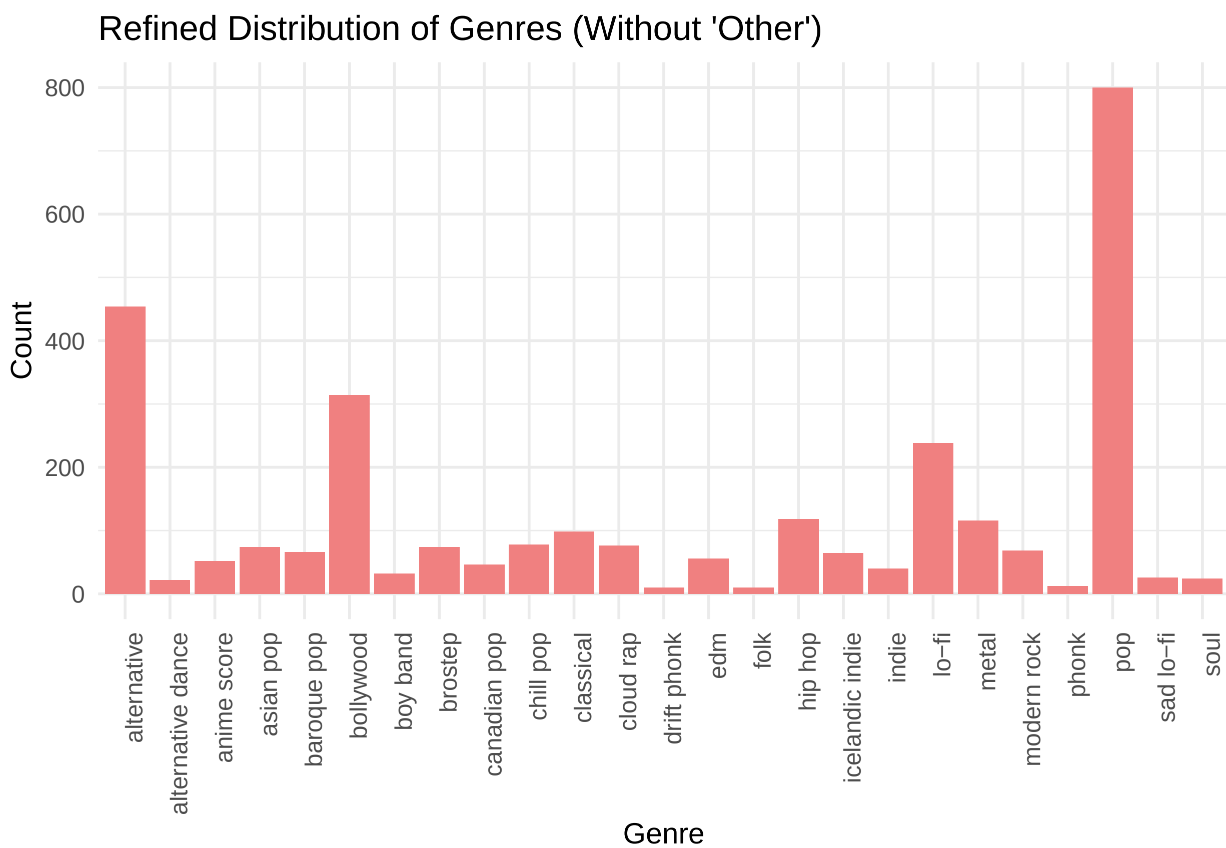
```r
summary(spotify_analysis_data_filtered)
```

```
##   trackName            msPlayed          danceability    danceability_imp
##  Length:2968        Min.   :        0   Min.   :0.0000   Mode :logical
##  Class :character   1st Qu.:   144138   1st Qu.:0.4014   FALSE:2968
##  Mode  :character   Median :   281007   Median :0.5399
##                     Mean   :  1732460   Mean   :0.5128
##                     3rd Qu.:  1141414   3rd Qu.:0.6553
##                     Max.   :158367130   Max.   :0.9852
##      energy       energy_imp          tempo        tempo_imp
##  Min.   :0.0000   Mode :logical   Min.   :0.0000   Mode :logical
##  1st Qu.:0.3815   FALSE:2968      1st Qu.:0.3555   FALSE:2968
##  Median :0.5601                   Median :0.4793
##  Mean   :0.5405                   Mean   :0.4936
##  3rd Qu.:0.7274                   3rd Qu.:0.6144
##  Max.   :0.9980                   Max.   :1.0000
##     genre             artistName
##  Length:2968        Length:2968
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

With the dataset filtered to exclude the "Other" category, we can now proceed with updated visualizations and analyses to explore the refined distributions and correlations among our key variables. This approach allows us to observe clearer patterns and insights without the ambiguity introduced by the mixed "Other" category.

Updated Visualizations for Refined Categorical Distributions - We will now generate updated visualizations to examine the distributions of `genre` and `artistName` in the filtered dataset. This will help confirm that the distributions are more meaningful and balanced for analysis.

```r
# Updated Genre Distribution after removing "Other"
ggplot(spotify_analysis_data_filtered, aes(x = genre)) +
  geom_bar(fill = "lightcoral") +
  xlab("Genre") +
  ylab("Count") +
  ggtitle("Refined Distribution of Genres (Without 'Other')") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

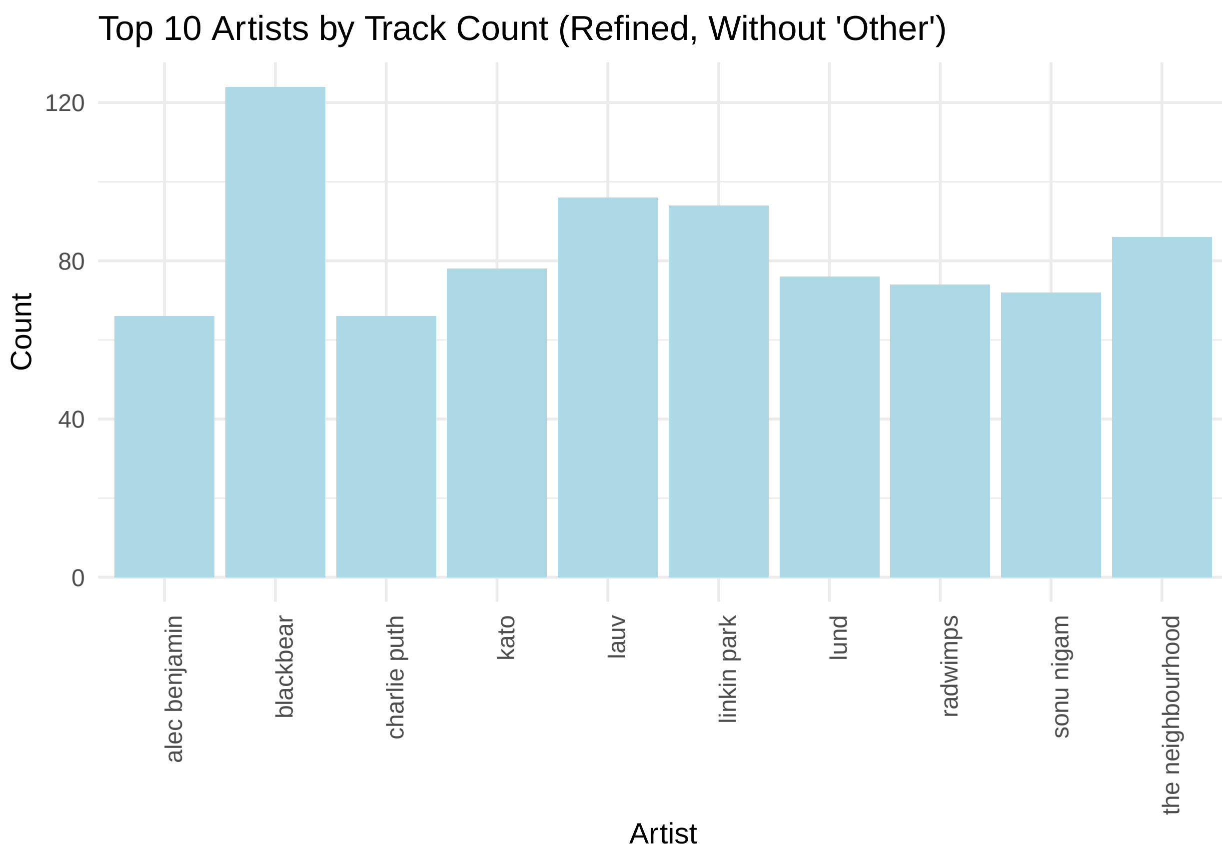


In this refined genre distribution (excluding "Other"), we observe:

- **Dominant Genres**: Pop and Phonk are the most represented, especially Phonk, suggesting a strong listener preference in the dataset.
- **Significant Counts**: Alternative and Bollywood also show substantial representation, hinting at their distinct listener base.
- **Niche Genres**: Genres like Soul and Sad Lo-fi have lower counts, adding diversity but limiting broader insights for these categories.

This distribution offers a clearer view without the "Other" category, allowing for more focused analysis.

```r
# Updated Artist Distribution after removing "Other"
# Filter for the top 10 artists by track count in the refined dataset
top_artists_filtered <- names(sort(table(spotify_analysis_data_filtered$artistName), decreasing = TRUE)
filtered_data_artist <- spotify_analysis_data_filtered %>%
  filter(artistName %in% top_artists_filtered)

ggplot(filtered_data_artist, aes(x = artistName)) +
  geom_bar(fill = "lightblue") +
  xlab("Artist") +
  ylab("Count") +
  ggtitle("Top 10 Artists by Track Count (Refined, Without 'Other')") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



This refined distribution of the top 10 artists (excluding "Other") reveals:

- **Most Popular Artist**: **Blackbear** leads significantly, followed by **Lauv** and **Linkin Park**, indicating these artists have a strong presence in the dataset.
- **Consistent Representation**: The remaining artists, including **Sonu Nigam** and The **Neighbourhood**, display relatively balanced track counts, suggesting a diversified listener interest.

This refined view removes ambiguity, allowing a clearer focus on individual artist impacts on playback trends. Next, we'll proceed to **Correlation Analysis** to evaluate how artist popularity may influence `msPlayed`.

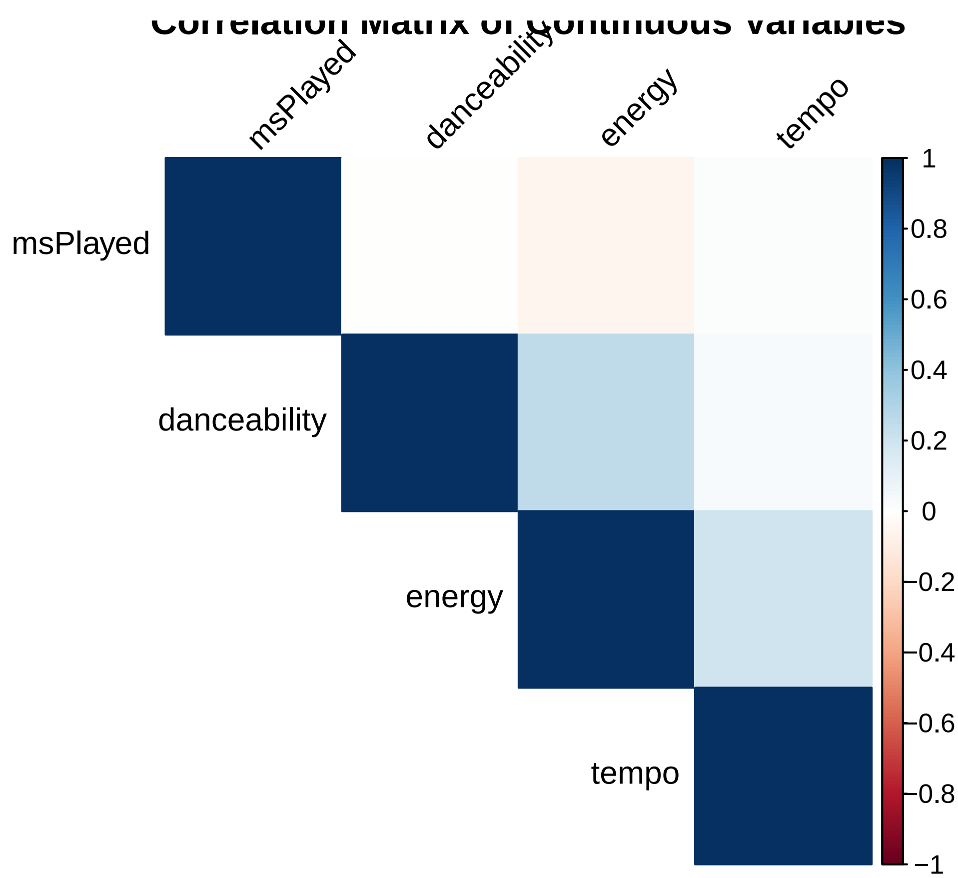
3.2 Correlation Analysis

3.2.1 Distribution of categorical variables (genre and artistName) With the refined dataset ready, we can proceed to **Correlation Analysis**. This step will help us identify relationships among continuous variables, especially their influence on the response variable, `msPlayed`. We aim to understand which features have stronger linear relationships, which will be helpful for subsequent modeling steps.

Let's move to computing and visualizing the correlation matrix for the continuous variables, focusing on identifying any potential predictors for `msPlayed`.

```r
# Select continuous variables for correlation analysis
continuous_data <- spotify_analysis_data_filtered %>% select(msPlayed, danceability, energy, tempo)

# Calculate correlation matrix
correlation_matrix <- cor(continuous_data, use = "complete.obs")
```

```r
# Plot the correlation matrix
corrplot(correlation_matrix, method = "color", type = "upper",
         title = "Correlation Matrix of Continuous Variables",
         tl.col = "black", tl.srt = 45)
```

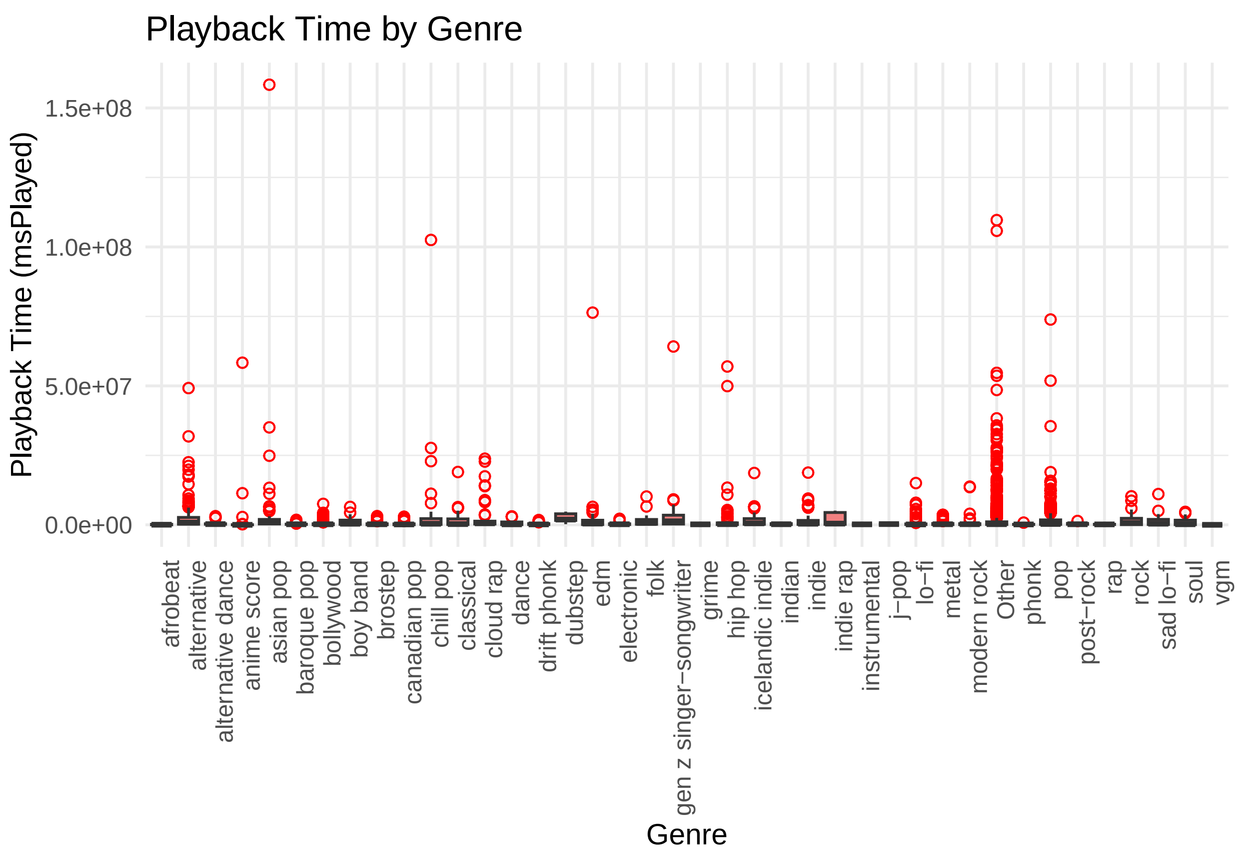


Interpretation of Correlation Matrix The correlation matrix reveals: - **`msPlayed` Correlations**: `msPlayed` shows weak correlations with `danceability`, `energy`, and `tempo`, indicating these variables alone don't strongly predict playback time. - **Inter-variable Relationships**: Moderate positive correlation exists between `danceability` and `energy`, while other pairs show low correlation, suggesting they vary independently.

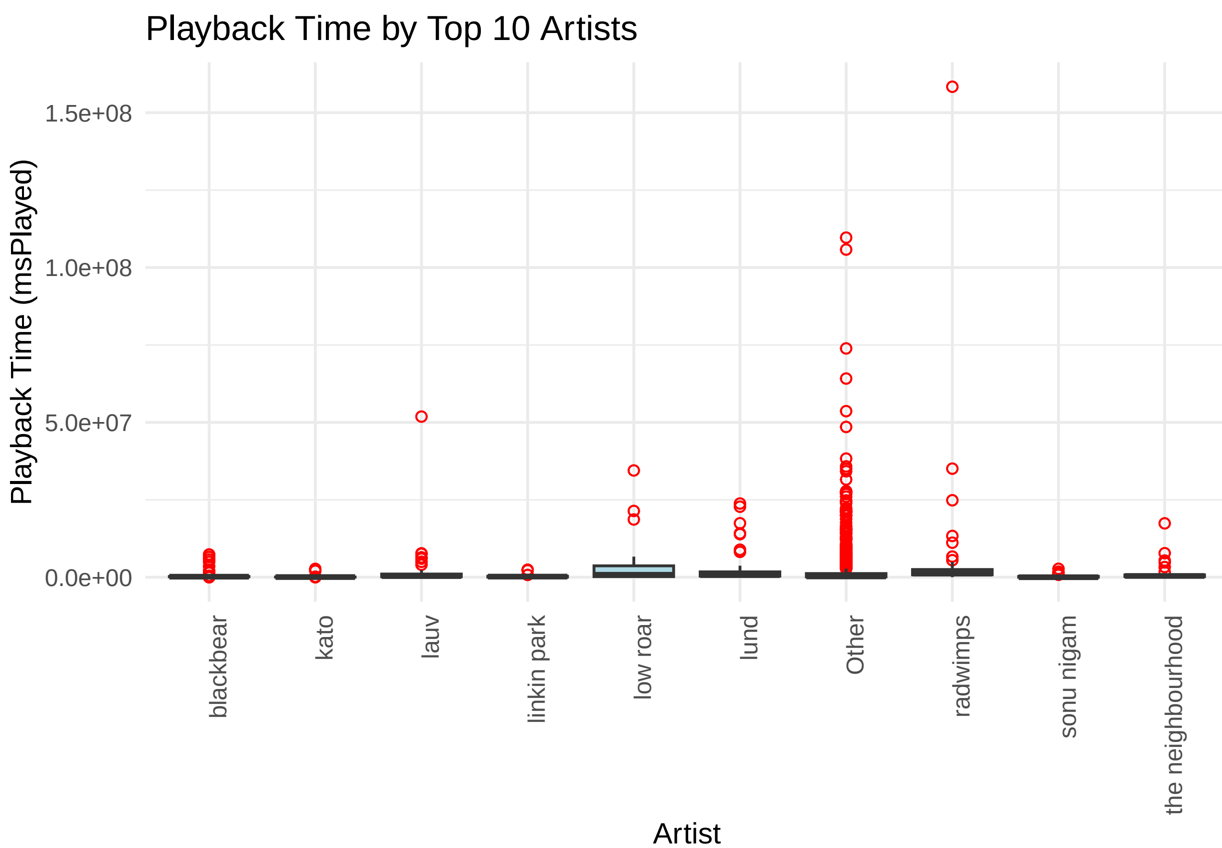
Given the weak continuous variable correlations, we'll: 1. **Examine Categorical Variables** (`genre` and `artistName`) to assess their impact on `msPlayed`. 2. **Visualize Categorical Influence** with box plots to compare playback time distributions across genres and artists.

3.2.2 Playback Time Analysis by Genre and Artist With the insights gained from the correlation matrix of continuous variables, we'll now explore the impact of categorical variables, specifically `genre` and `artistName`, on playback time (`msPlayed`). This step helps us assess whether these variables influence how long users listen to specific tracks, even if continuous predictors are weak. **Objective:** Use box plots to examine the influence of genre and artistName on msPlayed and identify patterns or outliers in listening duration.

```r
# Box Plot for `msPlayed` across different genres
ggplot(spotify_analysis_data, aes(x = genre, y = msPlayed)) +
  geom_boxplot(fill = "lightcoral", outlier.color = "red", outlier.shape = 1) +
  xlab("Genre") +
  ylab("Playback Time (msPlayed)") +
  ggtitle("Playback Time by Genre") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```r
# Box Plot for `msPlayed` across top artists
top_artists <- names(sort(table(spotify_analysis_data$artistName), decreasing = TRUE))[1:10]
filtered_data <- spotify_analysis_data %>% filter(artistName %in% top_artists)

ggplot(filtered_data, aes(x = artistName, y = msPlayed)) +
  geom_boxplot(fill = "lightblue", outlier.color = "red", outlier.shape = 1) +
  xlab("Artist") +
  ylab("Playback Time (msPlayed)") +
  ggtitle("Playback Time by Top 10 Artists") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



The **Playback Time by Genre** plot shows that genres like "alternative" and "pop" exhibit high variability in playback times, with several notable outliers, particularly in genres like "metal" and "pop." This suggests that certain tracks within these genres receive much higher engagement than others, potentially due to genre popularity or track-specific factors.

Similarly, the **Playback Time by Top 10 Artists** plot indicates that some artists, such as **"Low Roar,"** have tracks with particularly high playback times, standing out as potential favorites or frequently played songs. Meanwhile, popular artists like **"blackbear"** and **"the neighbourhood"** show more uniform playback times across their tracks, indicating consistent engagement levels among listeners.

While these insights from categorical variables provide an overview, a deeper exploration of **pairwise relationships between each predictor and playback time** (bivariate analysis) will help refine our understanding. This analysis will involve:

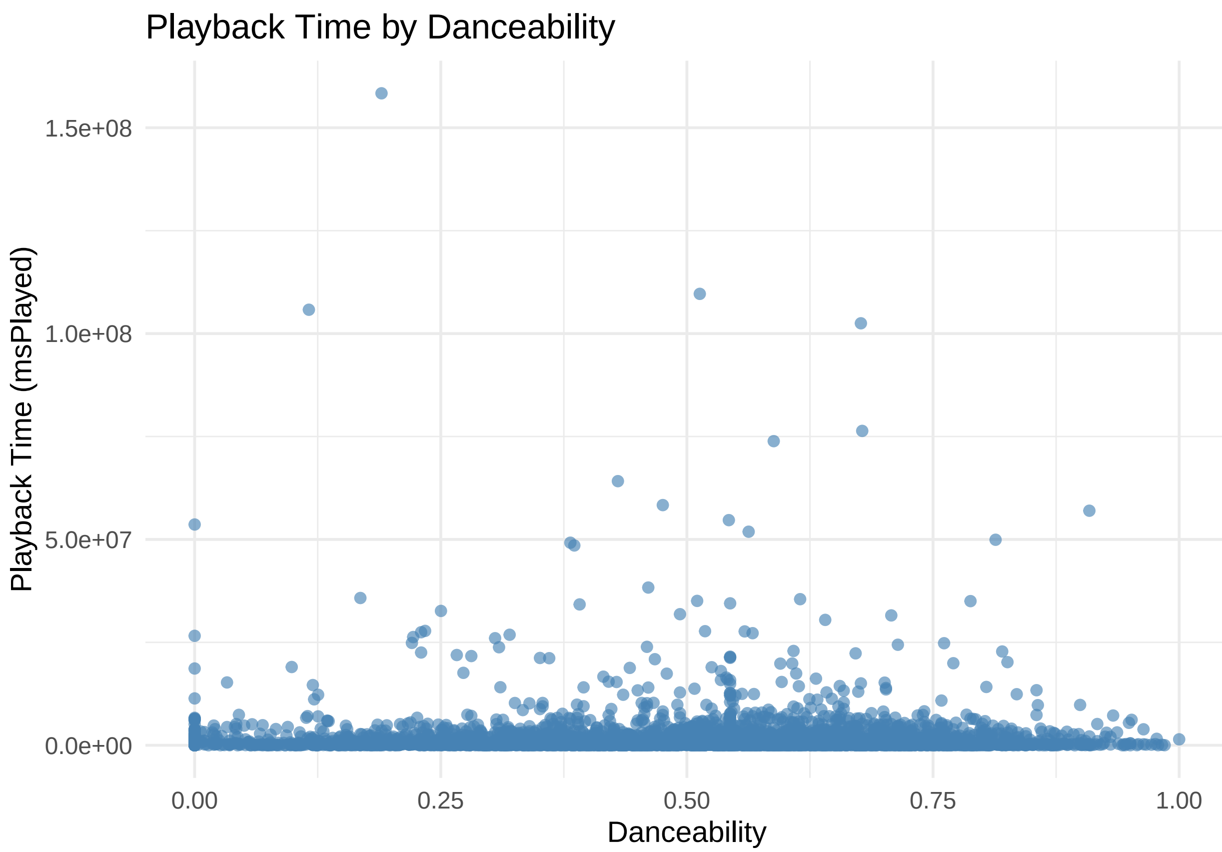
1. **Scatterplots** for continuous predictors (e.g., `danceability`, `energy`, `tempo`) against `msPlayed`, to observe any linear or non-linear relationships that may affect playback duration.
2. **Boxplots** for categorical variables (`genre` and `artistName`) against `msPlayed`, to investigate if playback time significantly varies across different categories.

These analyses will guide us in identifying relevant patterns and preparing for **Regression Modeling**, which aims to quantify and predict the impact of these features on playback time.

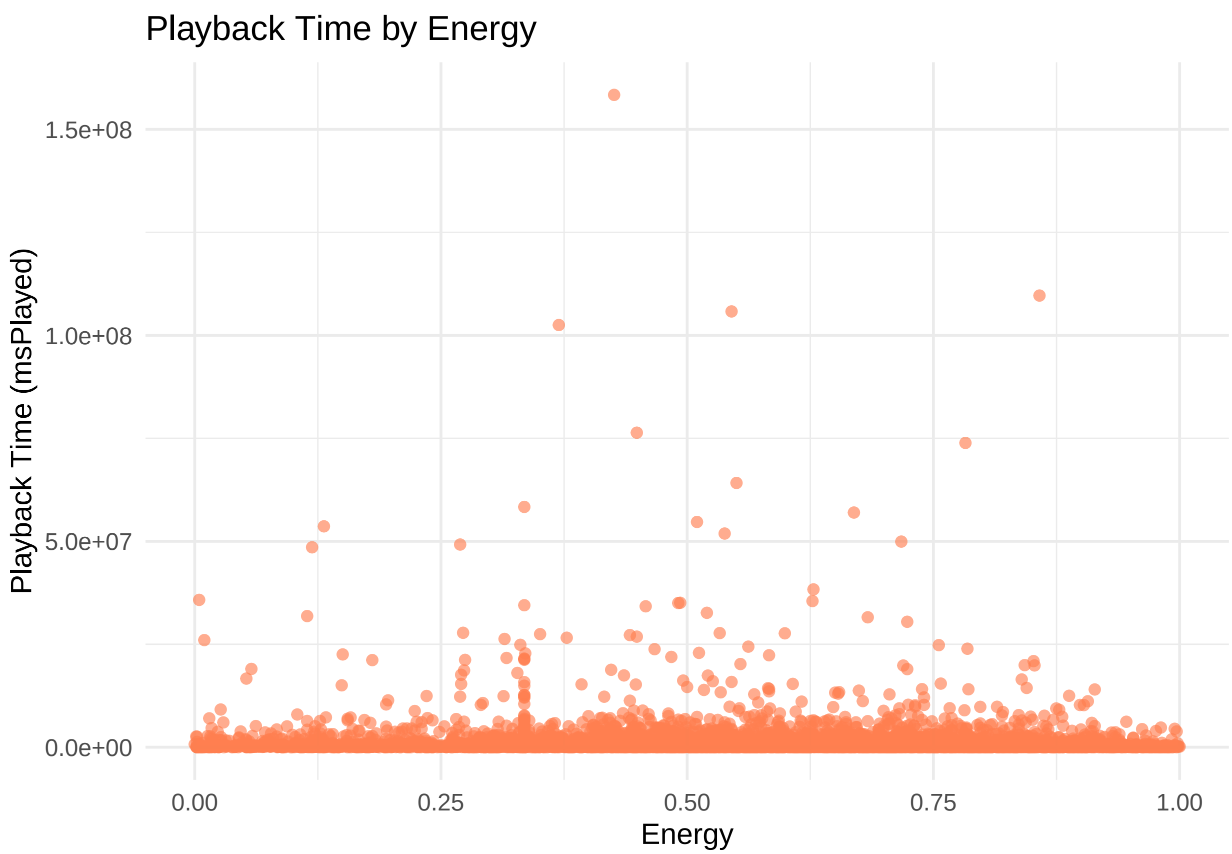
Let's proceed with the ### 4. Bivariate Analysis: #### 4.1 Scatterplots for Continuous Predictors

```
# Scatterplot for danceability vs. msPlayed
ggplot(spotify_analysis_data, aes(x = danceability, y = msPlayed)) +
  geom_point(alpha = 0.4, color = "steelblue") +
```

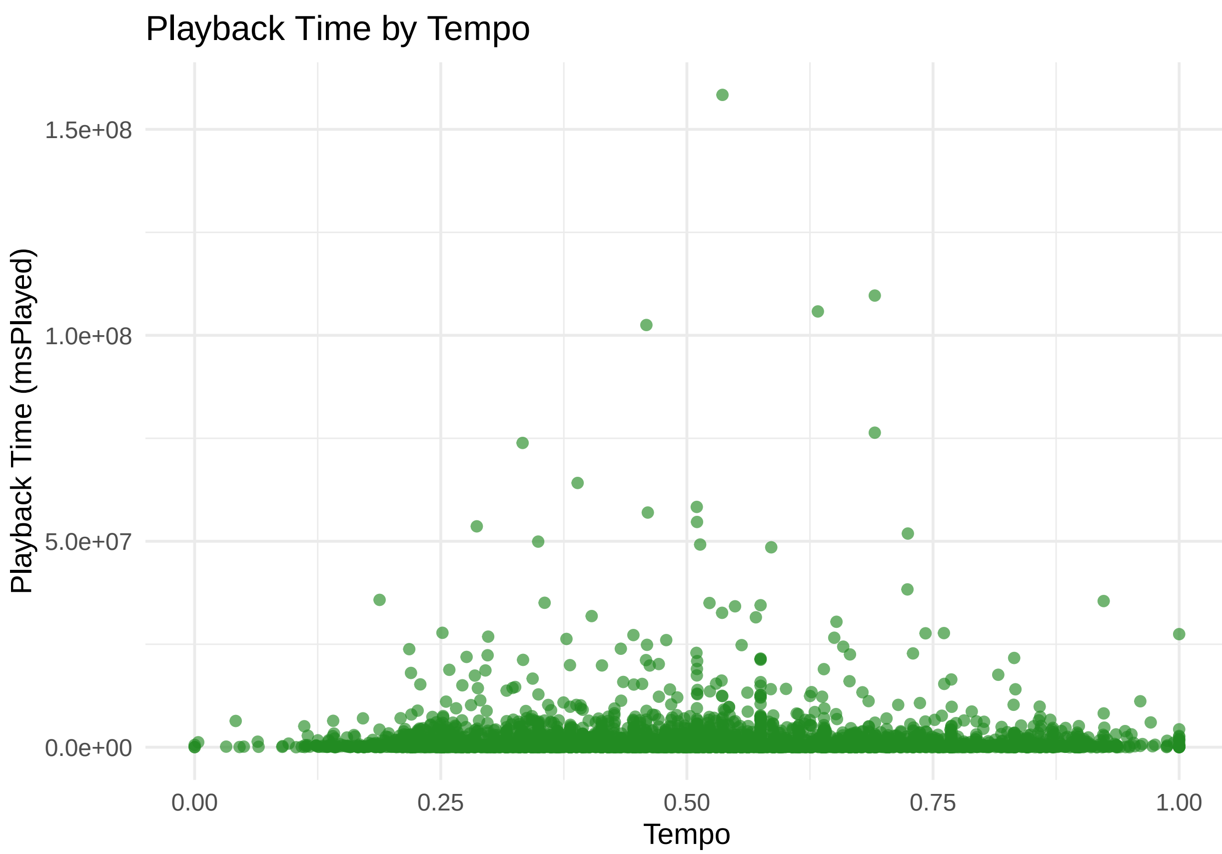
```r
  labs(title = "Playback Time by Danceability", x = "Danceability", y = "Playback Time (msPlayed)") +
  theme_minimal()
```



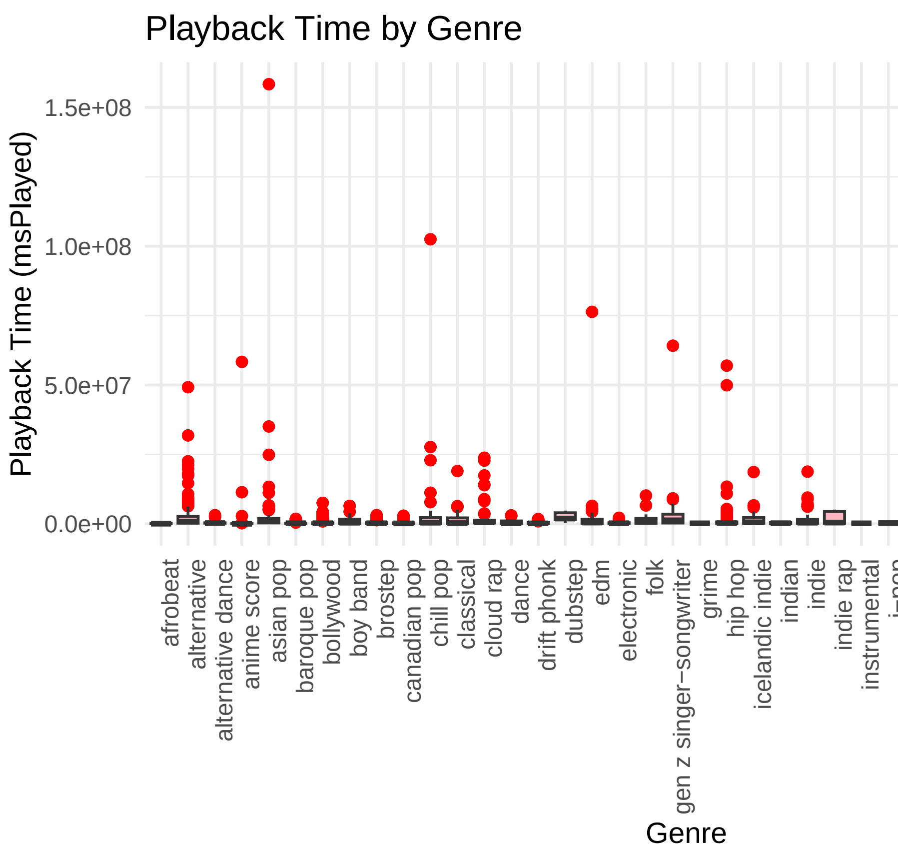

```r
# Scatterplot for energy vs. msPlayed
ggplot(spotify_analysis_data, aes(x = energy, y = msPlayed)) +
  geom_point(alpha = 0.4, color = "coral") +
  labs(title = "Playback Time by Energy", x = "Energy", y = "Playback Time (msPlayed)") +
  theme_minimal()
```




```r
# Scatterplot for tempo vs. msPlayed
ggplot(spotify_analysis_data, aes(x = tempo, y = msPlayed)) +
  geom_point(alpha = 0.4, color = "forestgreen") +
  labs(title = "Playback Time by Tempo", x = "Tempo", y = "Playback Time (msPlayed)") +
  theme_minimal()
```

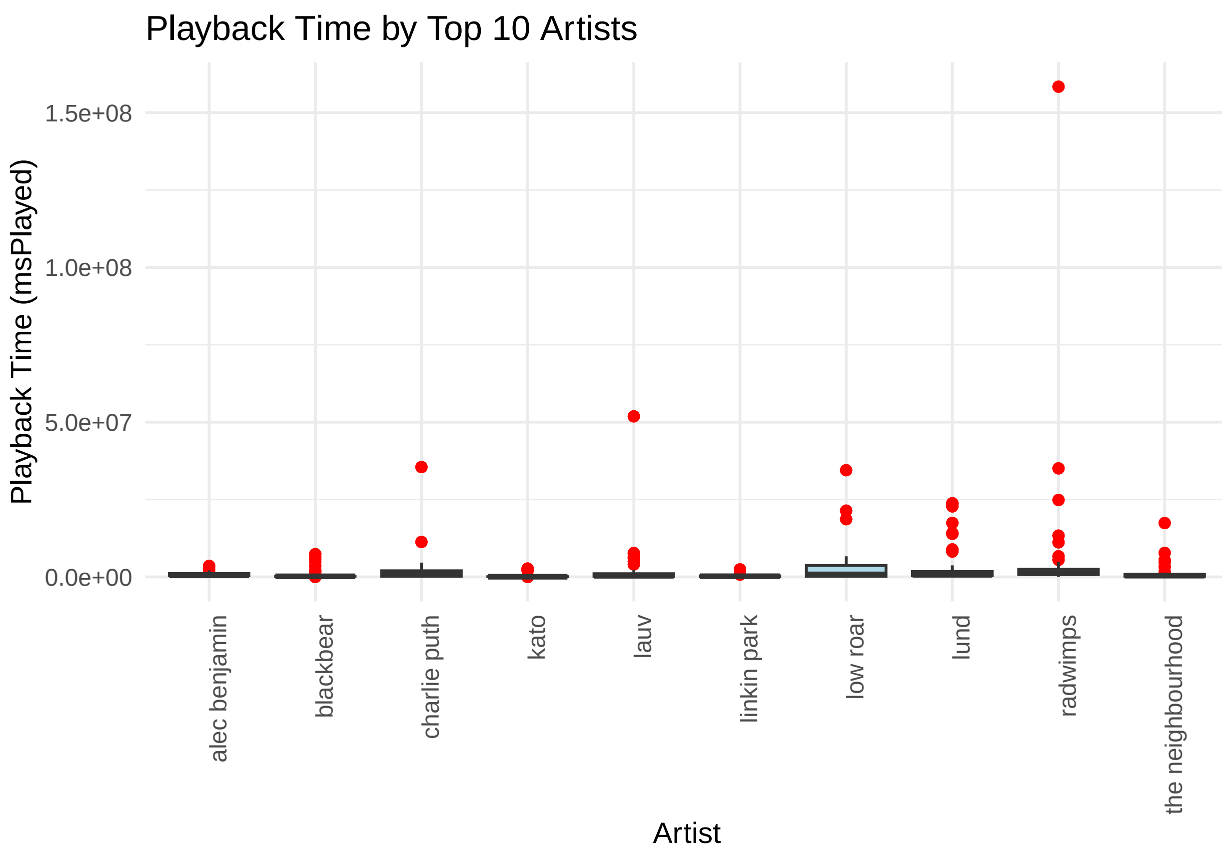



```r
# Boxplot for msPlayed by genre
ggplot(spotify_analysis_data, aes(x = genre, y = msPlayed)) +
  geom_boxplot(outlier.color = "red", fill = "lightpink") +
  labs(title = "Playback Time by Genre", x = "Genre", y = "Playback Time (msPlayed)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



4.2 Boxplots for Categorical Predictors

```r
# Boxplot for msPlayed by top 10 artists
# Filtering top 10 artists for boxplot
top_10_artists <- spotify_analysis_data %>%
  filter(artistName %in% c("blackbear", "alec benjamin", "charlie puth", "kato", "lauv",
                           "linkin park", "low roar", "lund", "radwimps", "the neighbourhood"))

ggplot(top_10_artists, aes(x = artistName, y = msPlayed)) +
  geom_boxplot(outlier.color = "red", fill = "lightblue") +
  labs(title = "Playback Time by Top 10 Artists", x = "Artist", y = "Playback Time (msPlayed)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



1. **Playback Time vs. Continuous Features**:
   - **Tempo**: The scatterplot shows no distinct pattern between tempo and playback time, with scattered data points across the range of tempo values. A few high outliers are evident, but no strong trend emerges.
   - **Energy**: Playback time does not show a clear relationship with energy levels. The distribution is uniform, suggesting energy might not be a significant predictor of playback time.
   - **Danceability**: While no evident trend links danceability to playback time, there are clusters of outliers for both low and high danceability values.
2. **Playback Time by Genre**:
   - Boxplots indicate that genres such as "metal" and "pop" have higher variability in playback times. Notable outliers exist, with some tracks receiving exceptionally high engagement.
   - Other genres like "classical" and "lo-fi" exhibit more consistent playback times with fewer extreme values.
3. **Playback Time by Top 10 Artists**:
   - Artists like **"Low Roar"** and **"Radwimps"** stand out with tracks having significantly high playback times, evident from the extreme outliers.
   - Popular artists like **"Blackbear"** and **"The Neighbourhood"** display more consistent playback times, indicating sustained and uniform engagement.

Key Insights:

- **No strong linear trends** exist between playback time and continuous variables (`tempo`, `energy`, `danceability`), though outliers suggest potential niche influences.

- **Genres and artists** exhibit distinctive playback time distributions, with some showing extreme variability, suggesting they might have niche, highly engaged audiences.

The variability observed, especially among genres and artists, suggests that categorical variables like `genre` and `artistName` might significantly influence playback time. The next logical step is to proceed with **Regression Modeling** to quantify the impact of these predictors and identify the key variables driving playback time.

3.3 Regression Modeling:

To understand the factors affecting playback time (`msPlayed`), we proceed to **Regression Modeling**. This phase aims to:

1. **Quantify Relationships**: Determine how continuous features (`danceability`, `energy`, `tempo`) and categorical features (`genre`, `artistName`) influence playback time.
2. **Predict Playback Duration**: Build a model to predict `msPlayed`, potentially revealing patterns for improved recommendations.
3. **Identify Key Predictors**: Pinpoint the most influential song attributes on playback duration.

Model Setup We'll use a **multiple linear regression** model where `msPlayed` is the dependent variable, and our predictors include `danceability`, `energy`, `tempo`, `genre`, and `artistName`. R will automatically create dummy variables for categorical predictors.

```r
# Fit the regression model
model <- lm(msPlayed ~ danceability + energy + tempo + genre + artistName, data = spotify_analysis_data_

# Summarize model results
summary(model)
```

```
##
## Call:
## lm(formula = msPlayed ~ danceability + energy + tempo + genre +
##     artistName, data = spotify_analysis_data_filtered)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -19780877  -1294276   -391396     88884 150717542
##
## Coefficients: (24 not defined because of singularities)
##                                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)                            1548411     956502   1.619   0.1056
## danceability                            894362     804036   1.112   0.2661
## energy                                 -941049     817744  -1.151   0.2499
## tempo                                   115152     714924   0.161   0.8721
## genrealternative dance                 -669171    2050069  -0.326   0.7441
## genreanime score                       1456964    1172415   1.243   0.2141
## genreasian pop                         6270391    1076029   5.827 6.27e-09 ***
## genrebaroque pop                      -1185334    1090181  -1.087   0.2770
## genrebollywood                        -1484480    2471031  -0.601   0.5481
## genreboy band                            75072    1590051   0.047   0.9623
## genrebrostep                          -1019048    1146760  -0.889   0.3743
## genrecanadian pop                      -823967    1650412  -0.499   0.6176
```

```
## genrechill pop                       4373950    1103751   3.963 7.59e-05 ***
## genreclassical                         68377    1949270   0.035   0.9720
## genrecloud rap                       1875101    1063809   1.763   0.0781 .
## genredrift phonk                     -970388    2237770  -0.434   0.6646
## genreedm                            19707874    2456438   8.023 1.50e-15 ***
## genrefolk                             124390    2217945   0.056   0.9553
## genrehip hop                         -540814    1666732  -0.324   0.7456
## genreicelandic indie                  704116    1133460   0.621   0.5345
## genreindie                            -26552    1487380  -0.018   0.9858
## genrelo-fi                          -1146366    1248018  -0.919   0.3584
## genremetal                            -99181    2054622  -0.048   0.9615
## genremodern rock                      160460    1322621   0.121   0.9034
## genrephonk                           -977231    2061320  -0.474   0.6355
## genrepop                              -70611    2220072  -0.032   0.9746
## genresad lo-fi                       -225667    2052210  -0.110   0.9124
## genresoul                             423464    1924180   0.220   0.8258
## artistNamea.r. rahman                 107104    2947229   0.036   0.9710
## artistNameaj mitchell                 321035    2048591   0.157   0.8755
## artistNamealan walker              -19995440    2944539  -6.791 1.35e-11 ***
## artistNamealec benjamin              -550541    1090894  -0.505   0.6138
## artistNamealice in chains            -500783    2849830  -0.176   0.8605
## artistNameanson seabra               1375316    2068581   0.665   0.5062
## artistNameap dhillon                 4582123    2052658   2.232   0.0257 *
## artistNamearijit singh                662887    3169077   0.209   0.8343
## artistNameastrid s                     43762    2217572   0.020   0.9843
## artistNameau/ra                       -46140    2221902  -0.021   0.9834
## artistNameaurora                       14505    2754163   0.005   0.9958
## artistNamebazzi                      -597597    2573558  -0.232   0.8164
## artistNameber                         222965    2058876   0.108   0.9138
## artistNamebesomorph                  -904785    2216154  -0.408   0.6831
## artistNamebillie eilish             -1101190    2989376  -0.368   0.7126
## artistNameblackbear                  -694905    2185548  -0.318   0.7505
## artistNameboy in space               1190384    1920407   0.620   0.5354
## artistNamebülow                       458104    2225716   0.206   0.8369
## artistNamecalvin harris              -989943    2849068  -0.347   0.7283
## artistNamecharlie puth                970718    2255238   0.430   0.6669
## artistNamecharlotte gainsbourg       -591264    2619880  -0.226   0.8215
## artistNamecharlotte lawrence         1000316    2225205   0.450   0.6531
## artistNamechris james                 854449    2851576   0.300   0.7645
## artistNamechristine and the queens  -1369538    2850501  -0.480   0.6309
## artistNamechromatics                 -641829    2843284  -0.226   0.8214
## artistNameclaude debussy            -1548268    2751007  -0.563   0.5736
## artistNameclean bandit              -1262129    2974133  -0.424   0.6713
## artistNamecolours in the dark        -241528    2138004  -0.113   0.9101
## artistNameconan gray                  908238    2677538   0.339   0.7345
## artistNameconnor price                -82916    2984600  -0.028   0.9778
## artistNamedaya                       2190163    2970739   0.737   0.4610
## artistNamediljit dosanjh              776836    3319074   0.234   0.8150
## artistNamedimension 32                395419    1757741   0.225   0.8220
## artistNamedj snake                 -20558853    3029251  -6.787 1.39e-11 ***
## artistNamedoja cat                  -1361850    2973211  -0.458   0.6470
## artistNameed sheeran                 -710660    2451668  -0.290   0.7719
## artistNameelley duhé                   36836    2052014   0.018   0.9857
## artistNameeminem                     2756145    1862857   1.480   0.1391
```

```
## artistNameenrique iglesias          2520879    2977865  0.847   0.3973
## artistNamefiji blue                -4493326    1959416 -2.293   0.0219 *
## artistNamefinneas                   3262936    1560659  2.091   0.0366 *
## artistNameflo rida                 -1245331    2971256 -0.419   0.6752
## artistNamefly by midnight            816271    2842510  0.287   0.7740
## artistNamegerardo millán            -498016    2134614 -0.233   0.8155
## artistNamegreen day                -1002040    2622975 -0.382   0.7025
## artistNamehans zimmer               1375463    1958771  0.702   0.4826
## artistNameharry styles              2858904    2967148  0.964   0.3354
## artistNamehiroyuki sawano                NA         NA     NA       NA
## artistNamehrvy                        24049    2679065  0.009   0.9928
## artistNameillenium                -19722636    3157487 -6.246 4.83e-10 ***
## artistNameimagine dragons          -1115748    1752969 -0.636   0.5245
## artistNamejack ü                  -19967690    3035331 -6.578 5.64e-11 ***
## artistNamejames arthur             -1130519    2753962 -0.411   0.6815
## artistNamejason mraz                1194470    2968971  0.402   0.6875
## artistNamejatin-lalit              -1048478    2752483 -0.381   0.7033
## artistNamejeremy zucker             -615302    1606646 -0.383   0.7018
## artistNamejohn k                    -430846    2677411 -0.161   0.8722
## artistNamejonas blue                -689150    2377730 -0.290   0.7720
## artistNamejordy chandra             -486386    2009554 -0.242   0.8088
## artistNamejosh golden               1970348    2446906  0.805   0.4207
## artistNamejp saxe                   -432884    2226756 -0.194   0.8459
## artistNamejulia michaels            -804544    2842486 -0.283   0.7772
## artistNamejustin bieber             -256655    1972307 -0.130   0.8965
## artistNamejvke                      1062824    2199096  0.483   0.6289
## artistNamekainbeats                 -597418    2299410 -0.260   0.7950
## artistNamekato                      -685501    1207718 -0.568   0.5704
## artistNamekhalid                    1279491    2971215  0.431   0.6668
## artistNamekina                      -310024    2611631 -0.119   0.9055
## artistNamekishore kumar              580855    3160858  0.184   0.8542
## artistNamekk                         553139    2512139  0.220   0.8257
## artistNamekute                           NA         NA     NA       NA
## artistNamekygo                           NA         NA     NA       NA
## artistNamekyu                       -442865    2303991 -0.192   0.8476
## artistNamelady gaga                -1077427    2974056 -0.362   0.7172
## artistNamelany                      1964992    2972581  0.661   0.5086
## artistNamelauv                       526024    2212745  0.238   0.8121
## artistNamelinearwave                -700686    2005122 -0.349   0.7268
## artistNamelinkin park               -819941    2034849 -0.403   0.6870
## artistNamelizzy mcalpine            3613249    2231131  1.619   0.1055
## artistNamelow roar                       NA         NA     NA       NA
## artistNameludwig van beethoven     -1662601    2745684 -0.606   0.5449
## artistNamelund                           NA         NA     NA       NA
## artistNamemajor lazer              -1218474    2974982 -0.410   0.6821
## artistNamemark ambor               -1150591    2840556 -0.405   0.6855
## artistNamemaroon 5                  -827144    2680930 -0.309   0.7577
## artistNamemarshmello                 114435    1880510  0.061   0.9515
## artistNamemc orsen                       NA         NA     NA       NA
## artistNamemokita                         NA         NA     NA       NA
## artistNamemunn                      1762554    2847073  0.619   0.5359
## artistNamenf                         515690    1828100  0.282   0.7779
## artistNamenic d                     -609699    2972529 -0.205   0.8375
## artistNamenickelback                     NA         NA     NA       NA
```

```
## artistNameolivia rodrigo               80508    2967206   0.027   0.9784
## artistNamepowfu                           NA         NA      NA       NA
## artistNamepritam                      458298    2521093   0.182   0.8558
## artistNamepyotr ilyich tchaikovsky        NA         NA      NA       NA
## artistNameradwimps                        NA         NA      NA       NA
## artistNameruel                        955988    1381602   0.692   0.4890
## artistNamesachin-jigar                170156    2950491   0.058   0.9540
## artistNamesam smith                  -434502    3149572  -0.138   0.8903
## artistNamesarcastic sounds            168111    1843435   0.091   0.9273
## artistNamesasha alex sloan           1482130    1126252   1.316   0.1883
## artistNameshankar-ehsaan-loy          345619    2784468   0.124   0.9012
## artistNameshawn mendes                    NA         NA      NA       NA
## artistNameshreya ghoshal              368168    3052469   0.121   0.9040
## artistNameskrillex                        NA         NA      NA       NA
## artistNamesody                        682432    2219939   0.307   0.7586
## artistNamesonu nigam                  263536    2481128   0.106   0.9154
## artistNamestephen                    1385625    2846038   0.487   0.6264
## artistNamesurfaces                  -1352752    2750885  -0.492   0.6229
## artistNametate mcrae                 1170082    1658386   0.706   0.4805
## artistNametaylor swift               -640502    2748179  -0.233   0.8157
## artistNamethe lumineers                   NA         NA      NA       NA
## artistNamethe neighbourhood               NA         NA      NA       NA
## artistNamethe presets                     NA         NA      NA       NA
## artistNametwenty one pilots               NA         NA      NA       NA
## artistNameuttam singh                 331632    2714198   0.122   0.9028
## artistNamevampire weekend                 NA         NA      NA       NA
## artistNamevishal-shekhar              393367    3322887   0.118   0.9058
## artistNamewhy don't we               -885415    2530533  -0.350   0.7264
## artistNamewillis                          NA         NA      NA       NA
## artistNamexxxtentacion                    NA         NA      NA       NA
## artistNameyasumu                          NA         NA      NA       NA
## artistNameyo yo honey singh               NA         NA      NA       NA
## artistNameyot club                        NA         NA      NA       NA
## artistNamezachary knowles                 NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6631000 on 2838 degrees of freedom
## Multiple R-squared:  0.0841, Adjusted R-squared:  0.04247
## F-statistic:  2.02 on 129 and 2838 DF,  p-value: 2.744e-10
```

Interpretation of Regression Results and Next Steps The regression results indicate a low R^2 (8.41%), meaning that our model explains only a small portion of the variability in playback time (`msPlayed`). This is expected given the complex and diverse nature of music listening behavior, which is influenced by many factors beyond song attributes.

Key Findings:

1. **Significant Predictors**:
   - **Genres**: Certain genres like **Asian Pop**, **Chill Pop**, and **EDM** show significant positive associations with playback time.
   - **Artists**: Specific artists, such as **Alan Walker**, **DJ Snake**, and **Illenium**, have significant coefficients, indicating their songs either consistently attract high or low playback times.

2. **Non-Significant Predictors**:
   - Many artists and genres have non-significant coefficients, suggesting that these categories do not substantially impact playback time in this model.
   - Continuous variables (`danceability`, `energy`, `tempo`) show low predictive power and insignificant coefficients, which aligns with the low correlations observed in our preliminary analysis.

Next Steps: Given these findings, we will: 1. **Evaluate Model Diagnostics**: Check for issues like multicollinearity and heteroscedasticity to ensure model robustness. 2. **Consider Alternative Models**: Explore non-linear models or machine learning techniques like decision trees or random forests, which might capture complex relationships better. 3. **Feature Engineering**: Create interaction terms or new categorical groupings to capture hidden patterns.

Let's proceed by analyzing model diagnostics to assess potential improvements.